Oftalmólogo 1.0

Memoria

Estudiante: Marc Perna Acevedo
Usuario en el campus: mpernaa
Consultor: Jordi Ceballos Villach
Estudios: E.T.I.G.
Curso: 2005/2

Resumen del proyecto

Oftalmólogo 1.0 es un software concebido para gestionar la consulta de un centro oftalmológico, cubre toda la operativa del centro desde el punto de vista del recepcionista, del oftalmólogo y como novedad, del paciente.

Oftalmólogo 1.0 trabaja con Internet de forma natural, por ejemplo, automáticamente envía emails de aviso a los pacientes que tienen que visitarse dentro de x días, de esta forma libera a los recepcionistas de la preocupación de llamar para recordar visitas en muchos casos y transmite al paciente sensación de eficacia y correcta gestión. También permite al paciente conectarse desde casa al centro y poder consultar su historial.

La dificultad de este proyecto no radica sólo en la gestión de la visitas, de los pacientes, de los oftalmólogos, de los usuarios, o del historial de un paciente, radica también en la preparación de una base consistente y pensada para soportar el crecimiento de la aplicación y para facilitar la vida al desarrollador.

El proyecto se ha basado en la plataforma .Net ya que es una plataforma joven, bien planteada, utiliza estándares y las últimas tecnologías, además existen multitud de proyectos que le dan soporte y por lo tanto le queda recorrido. El lenguaje que se ha utilizado es C#, por ser similar a Java y permitir programar con orientación a objetos (OOP – Object Oriented Programming).

Índice

Resumen del proyecto 2
Índice 3
1.- Introducción 4
1. 1.- Descripción y funcionamiento 4
1. 2.- Justif. TFC y contexto en el que se desarrolla 5
1. 3.- Objetivos del TFC 5
1. 4.- Enfoque y método utilizado 7
1. 5.- Planificación del proyecto 7
1. 6.- Productos obtenidos 8
1. 7.- Descripción de los capítulos siguientes 8
2.- Análisis de requisitos 10
2. 1.- Composición de la aplicación 10
2. 2.- Requisitos funcionales 10
2. 3.- Requisitos técnicos 11
2. 4.- Seguridad 11
3.- Arquitectura de la aplicación 12
3. 1.- Introducción 12
3. 2.- Descripción de las tablas 13
3. 3.- Seguridad 16
3. 4.- Configuración 18
4.- Diseño de la base de datos 19
4. 1.- Diagrama ER 19
5.- Diseño de las clases 20
5. 1.- Entidades empresariales 20
5. 2.- Clases de acceso a datos 20
5. 3.- Clases de la vista o GUI 26
5. 4.- Clases gestoras o del controlador 32
5. 5.- Paquetes 37
6.- Descripción de Oftalmólogo 1.0 38
6. 1.- Subsistema de conexión 38
6. 2.- Subsistema de seguridad o de administración 40
6. 3.- Subsistema de gestión 49
6. 4.- Subsistema del paciente 62
Conclusiones 64
Líneas futuras de trabajo 65
Bibliografía 66
Webs 66

1.- Introducción

1. 1.- Descripción y funcionamiento

Todo centro oftalmológico tiene telefonistas, oftalmólogos, pacientes y visitas. La gestión de todo ello puede llevarse manualmente, no obstante la rapidez y seguridad que aporta un programa especializado es muchísimo mayor.

El buen funcionamiento de un centro oftalmológico empieza en la recepción. Un telefonista recibe la llamada de un paciente, le pregunta si es su primera visita, caso positivo introduce sus datos, caso negativo localiza su ficha, a continuación le pregunta con que oftalmólogo y que día querría visitarse. Graba la visita, pasará entonces a estar introducida en la base de datos e irá cambiando de estados en función de su evolución. Los posibles estados de una visita son:

- No realizada. Todavía no ha llegado el día de la vista
- Presentado. El paciente se ha presentado y está esperando en la sala de espera.
- No se realizó. La visita no se realizó porque el paciente no se presentó.
- Se realizó. La visita se realizó.

Cuando llega el paciente al centro oftalmológico el recepcionista localiza su visita, la marca como “Presentado” y abre la ficha del paciente(no se visualizarán los datos de carácter médico, como enfermedades, operaciones,…) para confirmar que tiene todos los datos actualizados y completos, le preguntará si desea una contraseña (en caso de no disponer anteriormente de una) para acceder a través de la Web a su historial, en caso de desearlo y de que facilite su email se le envían los datos de acceso automáticamente (en caso de estar siendo dado de alta o en caso de pulsar el botón de envío si se están actualizando) a continuación pasa a la sala de espera.

El oftalmólogo tiene en pantalla la Gestión de Visitas (por defecto aparecen las visitas de hoy y sólo las suyas) en la que se le informa de las visitas y del estado en que se encuentran, se trata de la misma pantalla que recepción utiliza sólo que al oftalmólogo se le permite introducir el resultado de la visita y el tratamiento. El oftalmólogo llama a su siguiente visita y la hace pasar, marca la visita y comprueba los datos del paciente si lo desea (puede modificar tanto los datos personales como los médicos), realiza la visita y llama a la finalización de la misma, que es una pestaña donde introducirá los resultados.

Una vez haya entrado a la consulta el último paciente y se dé por cerrada el recepcionista llamará a un proceso de cierre del día, que marcará las visitas no realizadas como “no se realizó” y al mismo tiempo enviará emails a los pacientes que tengan consulta dentro de “x” días (o de un número prefijado o determinado de días) para recordarles su visita, a continuación extraerá un listado de las visitas previstas para dentro de “x” días (ese número prefijado o determinado de días) indicando los pacientes (DNI, nombre y apellidos, teléfono y su email), el usuario por tanto sabrá a que usuarios se les ha mandado email y a cuales no, para que les pueda llamar por teléfono.

Mediante una página Web del centro, el paciente podrá acceder consultar su historial, no verá sus datos personales ya que los conoce perfectamente, tan sólo verá el historial de sus visitas.

Un punto crítico de Oftalmólogo 1.0 es el tema de la “superposición de visitas” que se refiere a la coincidencia horaria de dos visitas. Tan sólo se comprobará que no

existan dos visitas que empiecen el mismo día a la misma hora para el mismo oftalmólogo, no se controlará un tiempo mínimo entre visitas ya que no puede determinarse.

1. 2.- Justificación del TFC y contexto en el que se desarrolla: punto de partida y aportación del TFC

Tuve la idea de poder desarrollar este proyecto el día que fui al oftalmólogo y observé sorprendido los siguientes hechos:

1. El médico le preguntaba a la recepcionista quién era el siguiente paciente, si la recepcionista atendía el teléfono el médico debía esperar. Me habría causado mejor impresión si directamente hubiera preguntado por mí sin consultar la recepcionista y por supuesto sin tenerse que esperar.
2. Me quedé perplejo cuando pasé a la consulta y el médico extrajo de un fichero mi expediente escrito en una ficha en soporte papel.
3. Me comentó que para la siguiente visita me acordara de no dilatarme los ojos, pensé, me acordaré? Y si antes de venir a la próxima pudiera saber que me dijo en la anterior? Estaría bien!
4. El diagnóstico y tratamiento de mi visita lo apuntó en esa misma ficha, eso sí, utilizando lápiz y goma.

Todos estos hechos me hicieron darle vueltas al problema y pensar como podría resolverse toda esta operativa con un software de gestión.

Consideré importante para el software que se llevara a cabo que:

- Proporcionara un cliente ágil basado en el escritorio para la gestión del centro por parte del personal
- Proporcionara los mecanismos oportunos para la protección y confidencialidad de los datos médicos de los pacientes.
- El hecho de enviar un email con información a los pacientes fuera absolutamente natural como parte lógica del proceso de gestión
- Se proporcionara al paciente una consulta de sus visitas para que pudiera ver que medicamentos se le recetaron en el pasado, que le diagnosticaron, que le comentó el oftalmólogo…. También sería útil para el paciente a la hora de ir a otro especialista ya que podría trasladar el historial.

No pude evitar irme de allí pensando todo lo que se podría hacer y lo fácil que sería para el médico y la recepcionista llevar todo el control con un software para la gestión de visitas médicas.

1. 3.- Objetivos del TFC

.Net supone una revolución, no tan sólo por las ideas que aporta (muchas propias, otras de Java, otras de Delphi…) sino porque entre otras muchas cosas:

- Ha sido capaz de unificar los lenguajes que Microsoft mantenía y generar un código intermedio, común a todos
- Ha sido capaz de unificar tipos de datos
- Ha acabado (en principio) con el infierno de las DLL (DLL Hell)
- Ha orientado a la Web los lenguajes
- Ha redefinido ASP a ASP .Net simplificándolo y aportando un nuevo punto de vista
- Ha aportado todo un marco de trabajo impresionantemente extenso

- Se ha abierto a estándares, WebServices, xml, xsl…
- Ha abierto la puerta sin quererlo (o no) a la aparición de frameworks para otras plataformas (proyecto Mono)

En este proyecto he querido basarme en diferentes áreas de .Net tocando lo más importante, no he querido centrarme tan sólo en un una y no ver las otras, he optado por aprender lo máximo posible, por eso he considerado muy importante trabajar en :

1. Orientación a objetos: Examinando el lenguaje, las particularidades del mismo, la sintaxis para trabajar la herencia, para definir propiedades, interficies, para heredar clases gráficas (herencia visual), para hacer clases abstractas, para implementar clases "no heredables" (sealed), …, en definitiva para sacar el máximo provecho al lenguaje en la programación orientada a objetos.

2. Conexión a datos. Entendiendo los mecanismos para tratar datos en una aplicación
   - Utilizando los diferentes objetos para realizar consultas o realizar operaciones
   - Entendiendo como .Net "cachea" parte de la base de datos en memoria (DataSet) y como puede sincronizarse posteriormente con la base de datos
   - Viendo las diferencias entre las clases de acceso a datos
   - Capturando errores específicos de SQL Server
   - Utilizando transacciones cuando se requería
   - Presentando datos en un informe de Crystal reports

3. WinForms: Examinando la potencia del lenguaje en el entorno de ventanas.
   - Haciendo el entorno lo más gráfico posible, iconos, alineaciones (Dock), estilos, fondos …
   - Trabajando sobre todo el objeto DataGrid que es el control más complejo, para aprender su utilización y conocer sus limitaciones, intentando personalizarlo al máximo (ocultando columnas, dibujando selecciones de filas completas, impidiendo inserciones, modificaciones, cambiando títulos de columnas, anchos, colores, actualizando datos, interceptando eventos de cambios…)
   - Usando el máximo número de controles gráficos posibles
   - Haciendo DataBindings de los controles (contra entidades empresariales)
   - Planteando el típico problema de los menús que pueden ser visibles de los que no en función del usuario
   - Jugando con los diferentes tipos de presentación de las ventanas, mostrándolas en modo MDI o modales (se acabaron los problemas de visual Basic de que una ventana modal no podía ser child y viceversa)

4. Asp.net: Examinando la potencia de esta tecnología para crear Websites y generar html para el cliente en función del navegador
   - Examinando lo fácil o difícil que es trabajar con sesiones
   - Examinando los eventos que suceden en las páginas Web
   - Entendiendo como se pueden presentar datos de una base de datos al usuario
   - Aprendiendo a validar la entrada de un usuario por un formulario personalizado (FormAuthentication)
   - Aprendiendo a utilizar validadores en la parte cliente y en la parte servidora, entendiendo como realizan su trabajo mediante JavaScript

5. Configuraciones de la aplicación:
   - Diseñando la configuración de la aplicación sobre ficheros Xml
   - Permitiendo formatear cadenas para establecer la conexión a la base de datos
   - Integrando los permisos de la aplicación con los de la base de datos
   - Aprendiendo de que partes consta un documento Xml, como se genera y como se lee

6. Web
   - Orientando la aplicación a la Web enviando mensajes tras determinadas acciones, en formato HTML
   - Abriendo la aplicación a la Web

1. 4.- Enfoque y método utilizado

El enfoque y método utilizado se ha basado en el sistema estándar de desarrollo de aplicaciones, análisis, diseño, implementación y pruebas.

La diferencia básica con un proyecto real ha sido el tiempo de desarrollo y se ha llegado a su conclusión como si de una cuenta atrás se tratara, por lo que el factor presión ha sido muy importante.

Otra diferencia con un proyecto real es que no ha existido un cliente que planteara sus problemas cotidianos, habría sido positivo ya que el software se habría enriquecido con sus aportaciones.

Oftalmólogo 1.0 se ha dividido en:

- **Subsistema de conexión**: Encargado de leer la configuración del cliente y gestionar la entrada en la aplicación del usuario y de activar los menús que serán visibles en base a su usuario y al grupo/s a el/los que pertenezca

- **Subsistema de seguridad o de administración**: Sólo accesible por el administrador, incluye las gestiones de grupos de usuarios, de usuarios y de asignaciones de usuarios a grupos, de grupos a usuarios, de menús a grupos, de menús a usuarios, excepcionalmente se ha incorporado la gestión de menús.

- **Subsistema de gestión**: Subsistema encargado de gestionar el centro oftalmológico en sí, pacientes, oftalmólogos y visitas.

- **Subsistema del paciente**: Subsistema encargado de dar acceso a los pacientes a un entorno Web en el que poder consultar su historial.

1. 5.- Planificación del proyecto

La planificación ha seguido fielmente los plazos establecidos por las pruebas de evaluación continuada (PACs) de la asignatura TFC dentro de los estudios de Ingeniera Técnica de Informática de Gestión (E.T.I.G.).

- PAC 1:
  - 26 de septiembre. Explicar el Proyecto y establecer la planificación.

- PAC 2:
  - 17 de octubre. Análisis (funcionalidades), y primera versión del prototipo (interfície gráfica).
  - 7 de noviembre. Diseño y prototipo definitivo.
- PAC 3:
  - 9 de diciembre. Implementación acabada 95% o 100%.
- Entrega definitiva:
  - 9 de enero. Memoria, PPT y la implementación definitiva.

Nombre de tarea	Duración	Comienzo	Fin
PAC 1 - Plan de trabajo	**12 días**	**jue 15/09/05**	**lun 26/09/05**
Establecer proyecto a desarrollar	7 días	jue 15/09/05	mié 21/09/05
Establecer planificación	4 días	jue 22/09/05	dom 25/09/05
Entrega	1 día	lun 26/09/05	lun 26/09/05
PAC 2 - Diseño y prototipo definitivo	**42 días**	**mar 27/09/05**	**lun 07/11/05**
Análisis y primera versión del prototipo	**21 días**	**mar 27/09/05**	**lun 17/10/05**
Documento de análisis	11 días	mar 27/09/05	vie 07/10/05
Primera versión del prototipo	13 días	mar 04/10/05	dom 16/10/05
Entrega	1 día	lun 17/10/05	lun 17/10/05
Diseño y prototipo definitivo	**20 días**	**mar 18/10/05**	**dom 06/11/05**
Documento de diseño	15 días	mar 18/10/05	mar 01/11/05
Prototipo definitivo	7 días	lun 31/10/05	dom 06/11/05
Entrega	1 día	lun 07/11/05	lun 07/11/05
PAC 3 - Implementación	**32 días**	**mar 08/11/05**	**vie 09/12/05**
Implementación entre 95%-100%	31 días	mar 08/11/05	jue 08/12/05
Entrega	1 día	vie 09/12/05	vie 09/12/05
Entrega Final - Memoria y PPT	**31 días**	**sáb 10/12/05**	**lun 09/01/06**
Implementación definitiva, corrección de bugs	11 días	sáb 10/12/05	mar 20/12/05
Memoria	7 días	mié 21/12/05	mar 27/12/05
PowerPoint	11 días	mié 28/12/05	sáb 07/01/06
Entrega	1 día	lun 09/01/06	lun 09/01/06

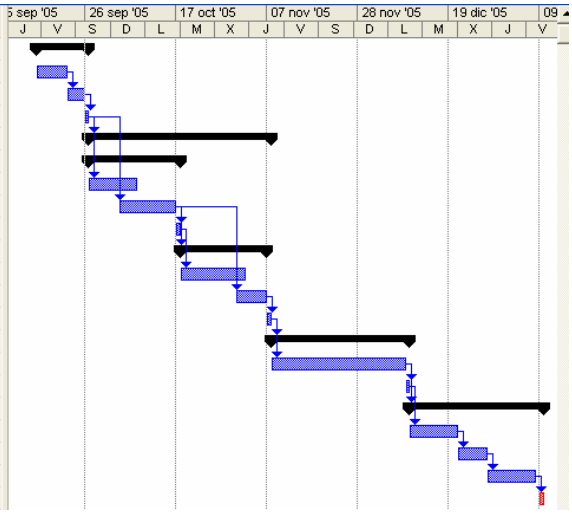


Diagrama de Gantt

1. 6.- Productos obtenidos

Durante el desarrollo del TFC se han obtenido los siguientes productos:

- Plan de trabajo
- Análisis
- Prototipo: Prototipo de la parte Windows en un proyecto WinForms y prototipo de la parte Web en páginas HTML estáticas
- Diagramas de clases (Entidades empresariales, clases de acceso a datos, clases de la vista y clases gestoras) y diagrama de la base de datos
- Implementación: Proyecto WinForms operativo, proyecto ASP .Net operativo y una biblioteca de clases común
- Manual de instalación
- Memoria del proyecto
- Presentación del proyecto

1. 7.- Descripción de los capítulos siguientes

Los próximos apartados forman un resumen del ciclo de la vida del software.

Apartado 2, Análisis de requisitos: Se identifican las metas globales, necesidades y requisitos del cliente, necesidades técnicas, nivel de seguridad requerido.

Apartado 3, Arquitectura de la aplicación: Se detalla la arquitectura de la aplicación, modelo utilizado, decisiones tomadas, seguridad y configuración.

Apartado 4, Diseño de la base de datos: Diagrama de datos con tablas y relaciones entre ellas.

Apartado 5, Diseño de las clases: Divisiones de clases, tipos de clases utilizadas, jerarquías, estructuración de la aplicación en paquetes…

Apartado 6, Descripción de Oftalmólogo 1.0. Descripción de la aplicación, principales funcionalidades, detalles, características de las pantallas, controles...

Finalmente se incluyen las conclusiones y las posibles mejoras del proyecto en próximas versiones.

2.- Análisis de requisitos

Es el primer paso del análisis del sistema, en este proceso el Analista se reúne con el usuario (un representante institucional, departamental o cliente particular) e identifican las metas globales, se analizan las perspectivas del cliente, sus necesidades y requerimientos sobre la planificación temporal y presupuestal y otros puntos que puedan ayudar a la identificación y desarrollo del proyecto.

No obstante este paso se ha realizado suponiendo los requisitos de un usuario hipotético. También se han incluido una serie de requisitos técnicos, a tener en cuenta para la documentación y desarrollo del ciclo de la vida del software.

2. 1.- Composición de la aplicación

La aplicación define divisiones en base a sus funcionalidades.

- **Subsistema de conexión**: Encargado de gestionar la entrada en la aplicación del usuario y de activar los menús que serán visibles en base a su usuario y al grupo/s a el/los que pertenezca

- **Subsistema de seguridad o de administración**: Sólo accesible por el administrador, incluye las gestiones de menús, de grupos de usuarios, de usuarios y de asignaciones de usuarios a grupos, de grupos a usuarios, de menús a grupos, de menús a usuarios.

- **Subsistema de gestión**: Subsistema encargado de gestionar el centro oftalmológico en sí, pacientes, oftalmólogos, visitas y cierre diario.

- **Subsistema del paciente**: Subsistema encargado de dar acceso a los pacientes a un entorno Web en el que podrán consultar su historial.

2. 2.- Requisitos funcionales

A partir de los datos aportados por el cliente se pueden extraer multitud de requisitos para la aplicación, detallarlos todos sería demasiado extenso, por eso citaré los más importantes, no obstante si se desean ver detalladamente se debería consultar el documento de Análisis.

Puesto que las funcionalidades se han agrupado en divisiones según su carácter las comentaré por subsistema.

Subsistema de conexión:
- Identificar el usuario
- Establecer su grupo/s y en base a los permisos que tenga (a nivel de usuario y o de grupo/s) visualizar los menús a los que podrá acceder

Subsistema de seguridad o de administración:
- Permitir la gestión de usuarios
- Permitir la gestión de grupos
- Permitir la gestión de menús (requisito para facilitar la tarea de añadir menús en la fase de desarrollo)

- Permitir establecer los miembros de un grupo o los grupos de un usuario
- Permitir establecer los menús accesibles para un grupo
- Permitir establecer los menús accesibles para un usuario, deberá poderse asignar o prohibir un menú exclusivamente a un usuario o dejarlo en base a la configuración de los grupos a los que pertenezca
- Permitir eliminar las visitas anteriores a una determinada fecha
- Permitir eliminar los pacientes sin visitas

Subsistema de gestión:

- Gestión de pacientes
- Gestión de visitas para recepcionista y oftalmólogos
- Cierre diario

Subsistema del paciente:

- Acceso al WebSite mediante contraseña
- Consulta de su historial

2. 3.- Requisitos técnicos

Como lenguaje de programación se utilizará C# tanto para la parte Windows como para la parte Web. La parte Web será una aplicación ASP .Net

Como sistema gestor de bases de datos se utilizará Microsoft SQL Server, y para acceder a datos se utilizará ADO .Net, concretamente las clases del espacio de nombres System.Data.SqlClient.

Como servidor Web se utilizará Microsoft Internet information Server.

El entorno de desarrollo será Microsoft Visual Studio 2003.

Para la elaboración de la documentación se utilizará Ms Word, Ms Project y para la elaboración de diagramas UML Ms Visio.

2. 4.- Seguridad

La seguridad es un punto muy importante, debe controlarse quien accede a la aplicación y delimitar claramente que puede hacer y que no ya que los datos médicos son confidenciales y es necesario tener un control máximo sobre ellos. Por eso es importante que determinados usuarios puedan ver tan sólo determinados menús, además el acceso a la base de datos debe ser lo más seguro posible.

3.- Arquitectura de la aplicación

3. 1.- Introducción

La forma de programar en estos años ha cambiado muchísimo, veamos algunos cambios importantes:

- Se han abandonado los entornos texto y ahora se exige que todas las aplicaciones(o casi todas) corran en entornos gráficos
- Los lenguajes de programación procedimentales han sido sustituidos por lenguajes orientados a objetos
- Se ha cambiado el análisis de las aplicaciones. Ahora usamos UML para analizarlas y la tendencia es sincronizar el código con los diagramas de UML, con lo que al final programaremos parte en código parte en diagramas (que generarán y/o actualizarán código).
- Se han incorporado eventos en las aplicaciones, también el concepto de hilos de ejecución
- Ha aumentado la necesidad de interoperabilidad entre lenguajes
- La programación se orientaba a un sistema operativo y ahora se tiende a poner una máquina virtual por medio para permitir que los programas sean multiplataforma o aislar los programas dentro de lo posible del sistema operativo
- Todo lenguaje se orienta a la Web y se le dota de los estándares y características imprescindibles
- Los lenguajes ahora prefieren seguir estándares abiertos a ser particularistas
- La jerarquía de clases de los lenguajes, crece y crece aportando más y más clases a los desarrolladores
- Ha aumentado la necesidad de crear aplicaciones seguras, no sólo porque los usuarios tengan más conocimientos, también porque las aplicaciones quedan expuestas en la red de redes y por tanto son susceptibles de ser atacadas
- Hemos pasado de utilizar editores (como el edit de ms-dos) o entornos de programación en modo texto como Turbo C o Turbo Pascal a utilizar unos IDEs muy completos, cargados de asistentes y ayudas para la programación.
- Los controles de las aplicaciones se han sofisticado. De trabajar en modo texto con campos con formato ahora disponemos de multitud de cajas de texto (de sólo texto, con calendarios, con calculadoras, con flechas arriba-abajo,…). Y no hablemos de cómo han cambiado las rejillas…, en definitiva, los controles se han sofisticado y han aparecido muchísimo nuevos

Programar, cada día es más complicado porque las exigencias aumentan y los conocimientos técnicos se amplían, para ayudarnos los IDEs proporcionan multitud de plantillas, de asistentes y facilidades, pero tanta facilidad también presenta inconvenientes ya que cada vez que utilizamos asistentes, plantillas, … se genera código.

También el programador tiende a llevar a cabo prácticas inadecuadas, por ejemplo en los formularios tiende a utilizar varios asistentes y acaba escribiendo código que lo hace todo, es decir, que accede a datos, que valida, que lleva la lógica de comprobación, que lleva la presentación…, los anglosajones denominan a estas pantallas “Quick-and-dirty GUIs”.

Los inconvenientes de mezclarlo todo son:

- Dificultad para mantener

- Código redundante en diferentes pantallas
- Dificultad de testeo
- Diseño en una única dirección, desde el GUI a la capa de negocios – que pasa si los objetos de negocio necesitan actualizar el GUI?

Por tanto se hace necesario dividir las pantallas para establecer responsabilidades de forma clara, "divide y vencerás" y por eso esta aplicación ha utilizado el modelo MVC.

3. 2.- Modelo MVC

El principio de división que va a seguir este proyecto es:

1. **Componentes de entidad empresarial:** Los componentes de entidad empresarial se utilizan para representar entidades del mundo real, como los oftalmólogos o los pacientes. Hay diversos métodos para implementar estas entidades empresariales, más adelante se tratará este punto

2. **Componentes de Acceso a datos:** Encargados de extraer y guardar entidades empresariales en la base de datos. Contendrán la lógica empresarial necesaria para alcanzar las operaciones relacionadas con los datos.

3. **Presentación de datos**: La interfície en sí, el GUI. Proporciona al usuario la información tanto para visualizarla como para introducirla.

4. **Gestores**: Encargados de mover y hacer cooperar a los componentes anteriores

Normalmente estas divisiones se agrupan en tres:

1. **Modelo**: Incluye la entidad empresarial y el acceso a datos

2. **Vista**: El formulario o la página ASP .Net

3. **Controlador**: El Gestor

Y forman un patrón de diseño denominado MVC (Model, View, Controller), a su vez existen múltiples estrategias de implementación de dicho modelo, Observer, Command…

Componentes de acceso a datos

Un componente de acceso a datos ofrece métodos para llevar a cabo operaciones sobre la base de datos, su misión es:

- Crear registros, sería guardar entidades
- Leer registros, sería devolver entidades
- Modificar registros con los datos de las entidades, ya revisados y que proporciona el llamador
- Eliminar registros, eliminar entidades o parte de ellas

A estos componentes los llamaremos CRUD (**C**reate, **R**ead, **U**pdate, **D**elete), por ello en la aplicación veremos diferentes clases cuyo nombre empezará por CRUD.

Normalmente un CRUD encapsula operaciones relacionadas con una única tabla o un grupo de ellas relacionadas (relaciones simples, relaciones master detail, relaciones muchos a muchos).

Los CRUDs que se han implementado trabajan sobre la rama de acceso a datos System.Data.SqlClient ya que el sistema gestor de base de datos que hemos utilizado es SQL Server.

Los componentes de acceso a datos recibirán como parámetros de entrada salida entidades empresariales o DataSet para pasar colecciones, en algunos casos.

Puesto que los CRUDs deberán ofrecer operaciones muy similares, se diseñará una jerarquía de clases que cubra todas las necesidades de este proyecto.

Componentes de entidad empresarial

Los componentes de entidad empresarial están desvinculados de los datos, lo que provoca una independencia total de su serialización / desserialización, de forma que la reutilización de código se garantiza en caso por ejemplo, de cambio de sistema gestor de base de datos.

Este proyecto presenta las siguientes necesidades con las entidades empresariales:

- Se necesitan enlazar a Windows Forms y en algún caso a páginas ASP .Net
- No se necesitan operaciones de ordenación a priori
- Unas veces se trataran individualmente mientras que otras veces en grupo
- La aplicación se implementará de forma local, no es necesario transmitirlas
- No se utilizarán servicios Web

Existen diferentes formas de representar una entidad empresarial

- XML
- Conjunto de datos genérico
- Conjunto de datos con tipo
- Entidad empresarial personalizada
- Entidad empresarial personalizada con comportamientos CRUD

Hay determinadas entidades que no necesitarán ser instanciadas nunca individualmente, por lo que no serán implementadas, por ejemplo la clase Menú. No obstante se necesitará recuperar colecciones de Menús, se empleará en estos casos DataSet. Por ello si existirá un CRUDMenú, no una entidad empresarial Menú.

En este proyecto se opta por el uso de entidades empresariales personalizadas (EEP), contienen normalmente los siguientes miembros:

- Campos privados para almacenar de forma local los datos de la entidad empresarial. Estos campos contienen una instantánea de los datos de la base de datos del momento en que fueron recuperados por el componente de acceso a datos.
- Propiedades públicas para obtener acceso al estado de la entidad y a las subcolecciones y jerarquías de los datos dentro de la entidad.
- Métodos y propiedades.
- Eventos para señalar los cambios en el estado interno del componente de la entidad. Muy útiles en los GUI.

Las EEP aportan las siguientes ventajas:

- Legibilidad del código (al no mezclar la entidad, por ejemplo, con el acceso a datos)
- Encapsulación
- Modelado de sistemas complejos
- Validación localizada
- Campos privados

Pero también tienen sus inconvenientes:

- Colecciones de entidades empresariales. Una EEP representa una sola entidad empresarial, no una colección de entidades empresariales. La aplicación de llamada debe crear una matriz o una colección de .NET para alojar varias entidades empresariales. Superable utilizando DataSet.
- Representación de jerarquías y relaciones complejas en una entidad empresarial. Para solucionarlo la entidad empresarial puede tener un DataSet.
- Búsqueda y ordenación de datos. Se puede implementar IComparable para solucionarlo.
- Problemas de extensibilidad. Si se modifica el esquema de la base de datos, tal vez se deba modificar la clase de la entidad personalizada y volver a implementar el ensamblado. No evitable.

Si las EEP se van a utilizar en interfaces de usuario y deseamos aprovechar el enlace de datos automático, puede que necesitemos implementar el enlace de datos en la EEP:

- Enlace de datos en Windows Forms. Puede enlazar los datos de una instancia de entidad con los controles sin implementar las interfaces de enlace de datos en la entidad personalizada. También se pueden enlazar los datos de una matriz o de una colección .NET de entidades.
- Enlace de datos en Web Forms. No se pueden enlazar datos de una instancia de entidad con los controles en un formulario Web sin implementar la interfaz IBindingList. Sin embargo, si se desea enlazar únicamente datos de conjuntos, se puede usar una matriz o una colección .NET sin tener que implementar la interfaz IBindingList en la entidad personalizada.

Presentación de datos

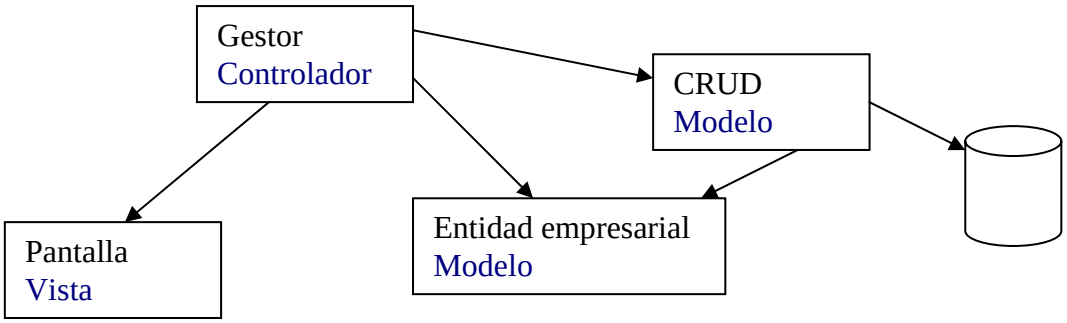
La aplicación esta divida en dos presentaciones de datos diferenciadas:

- Windows Forms
- Páginas ASP.Net

Tal como se ha visto en el apartado anterior el enlace a Windows Forms no presenta problemas, no obstante, no sucede lo mismo con ASP .Net ya que en Web Forms no se pueden enlazar datos de una instancia de entidad con los controles en un formulario Web sin implementar la interfaz IBindingList.

Gestores

El trabajo de las pantallas con sus respectivas entidades empresariales y con su acceso a datos dependerá de un gestor



En algunos diseños se permite a la pantalla acceder a la entidad empresarial pero nunca a la entidad empresarial acceder a la pantalla.

Se puede observar como la presentación de datos va por un lado, la entidad empresarial y la lógica por otro y el acceso a datos por otro, es el gestor el encargado de coordinarlo y enlazarlo todo convenientemente.

Cada parte está perfectamente dividida, Presentación e introducción de datos -> Pantalla, Lógica de negocio -> Entidad Empresarial,
Acceso a datos -> CRUD, Coordinación y enlaces necesarios -> Gestor.

Es importante seguir siempre la misma metodología de trabajo, de esta forma nos habituaremos a seguir una serie de pautas no sólo para crear una nueva pantalla, además para extenderla posteriormente, lo que repercutirá en hacer nuestro código lo más homogéneo posible, y por tanto, más claro y fácil de mantener.

3. 3.- Seguridad

La seguridad está compuesta por 2 niveles:

- Nivel 1, motor de la base de datos: No podrá acceder ningún usuario a la aplicación si no existe como inicio de sesión de SQL Server y no podrá llevar a cabo las operaciones que desde el motor se le prohíban
- Nivel 2, seguridad de la aplicación: La seguridad de la aplicación estará basada en usuarios y grupos de usuarios, a dichos objetos se les podrán asignar permisos para ver determinados menús o también para no ver determinadas partes de una pantalla (Recordemos que los recepcionistas no pueden ver los datos médicos de los pacientes)

Se ha implementado una gestión de usuarios (consulta, alta, baja y modificación) pero a diferencia de otros programas se han ligado los usuarios propios del software con los usuarios propios del motor de la base de datos, en este caso con los inicios de sesión de SQL Server, por ello:

1. Cuando se crea un usuario de la aplicación, transparentemente se crea un inicio de sesión de SQL Server y se le asignan los permisos oportunos a la base de datos
2. Cuando se modifica un usuario también se modifica el inicio de sesión de SQL Server (cambio de contraseña)
3. Cuando se elimina un usuario se pregunta si también debe hacerse de SQL Server (como mínimo siempre se quitará de la base de datos pero puede no quererse eliminar de los inicios de sesión de SQL Server porque a lo mejor el usuario utiliza otras bases de datos)

Para gestionar los permisos de los usuarios de forma colectiva y por tanto con mucha más facilidad se crearán grupos de usuarios, tan sólo a efectos de la aplicación. Un usuario podrá pertenecer a varios grupos simultáneamente.

Oftalmólogo 1.0 por defecto incorpora los usuarios:

- Admin (contraseña admin), con todos los permisos necesarios, es el usuario responsable de gestionar todos los aspectos de seguridad de la aplicación. No se puede eliminar este usuario en ningún caso.
- Recep (contraseña recep), con todos los permisos necesarios, se trata de un recepcionista
- Oftal (contraseña oftal), con todos los permisos necesarios, se trata de un oftalmólogo, si no hubiera ninguno la pantalla de vistas se quejaría ya que no podrían asignarse visitas.

Y los siguientes grupos:

- Administradores: Grupo de los usuarios administradores de la aplicación. No puede eliminarse este grupo en ningún caso.
- Oftalmólogos: Grupo de los oftalmólogos. Los usuarios que pertenezcan a este grupo gozarán del privilegio de ver los datos médicos de la ficha de pacientes y los resultados y tratamientos de las visitas. No puede eliminarse este grupo en ningún caso.
- Recepcionistas: Grupo de los recepcionistas de la aplicación.

Puesto que tanto los recepcionistas como los oftalmólogos son usuarios de la aplicación lo único que los diferenciará será el grupo al que pertenezcan.

**Se aporta la gestión de menús (consulta, alta, baja y modificación de menús), quiero puntualizar que esta pantalla sólo debería ser visible para el desarrollador, no obstante se incluye excepcionalmente en el perfil del administrador, si realmente la aplicación se instalara en un centro, el menú de Gestión de menús no sería accesible para nadie.

Desde el punto de vista de un proyecto de la extensión y complejidad de Oftalmólogo 1.0 seguramente no habría sido necesario controlar grupos de usuarios y visualización de menús de una forma tan configurable como la que se ha llevado a cabo, pero se ha hecho porque Oftalmólogo 1.0, es en definitiva la versión 1.0 y puede crecer con el tiempo y está pensado para crecer, por ello aporta grupos de usuarios, usuarios y permisos sobre menús a nivel de usuarios, de grupos e incluso de grupos y usuarios (OR de los permisos de los grupos de los que es miembro un usuario junto con los suyos propios)

Cuando un usuario accede a la aplicación se le pide su nombre de usuario y contraseña, se incluyen en la cadena de conexión y de esta forma evitamos dejar ficheros de configuración con un usuario genérico para la conexión a la base de datos y con una contraseña fácilmente localizable, reforzamos el sistema de seguridad de la aplicación y delegamos responsabilidades al motor de la base de datos.

3. 4.- Configuración

Lado cliente (aplicación WinForms)

Puede localizarse el fichero Config.xml en la misma ruta que el archivo OftalWinApp.exe

Existen diferentes apartados que resumiré muy brevemente (para ampliar esta información puede consultarse la guía de instalación que adjunto con la entrega):

- DBConnection: Cadena de conexión a SQL Server. {0} y {1} serán reemplazados por usuario y contraseña del cliente que trate de conectarse
- Smtp: Se configurará aquí la conexión al servidor smtp para el envío de emails.
- De, Asunto, Cuerpo: Configuración del email que será enviado al paciente para recordarle su próxima visita
- WebSite: URL del website de la consulta desde el que el paciente puede conectarse
- CuerpoAcceso: Mensaje para informar al usuario de su clave de acceso a nuestro WebSite, y la dirección del mismo.

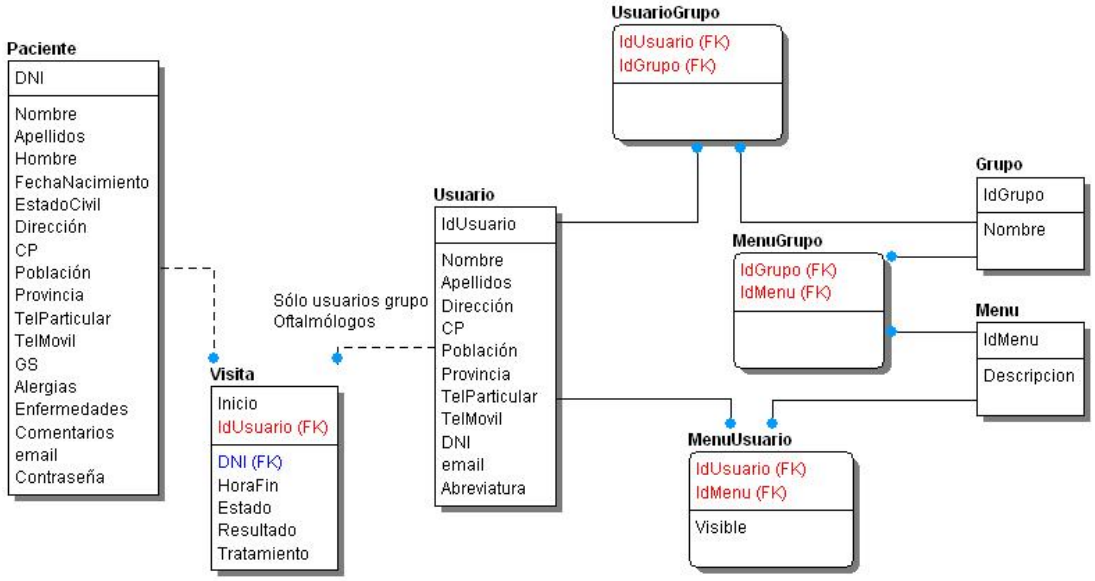
Servidor Web (aplicación ASP .Net)

La parte Web se configura mediante el fichero Config.XML, ubicado en la carpeta bin de la aplicación. En dicho fichero debe incluirse la cadena de conexión al servidor SQL, también debe proveerse un usuario y contraseña válidos.

4.- Diseño de la base de datos

4. 1.- Diagrama ER (Entity Relation)

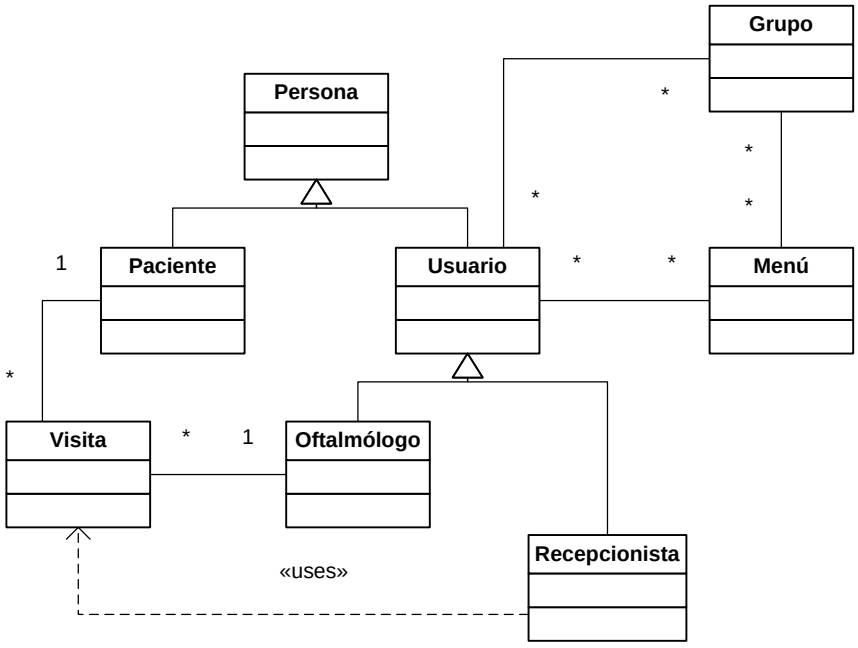
Se trata del diagrama de datos, debe aportar la infraestructura suficiente para almacenar y tratar las entidades empresariales de la aplicación que deban ser persistentes.



5.- Diseño de las clases

Vamos a separar los diagramas siguiendo el punto 3.- Diseño de la arquitectura de la aplicación, ya que si quisiéramos representar todas las clases en un solo diagrama no nos ayudaría a entender la estructuración de la aplicación.

5. 1.- Entidades empresariales



Las entidades tendrán las propiedades correspondientes que permitirán establecer y/o obtener sus valores.

También tendrán los métodos necesarios que permitirán que efectúen las operaciones que deban efectuar.

Como mínimo tendrán un constructor, e incluirá los parámetros necesarios para inicializar la instancia.

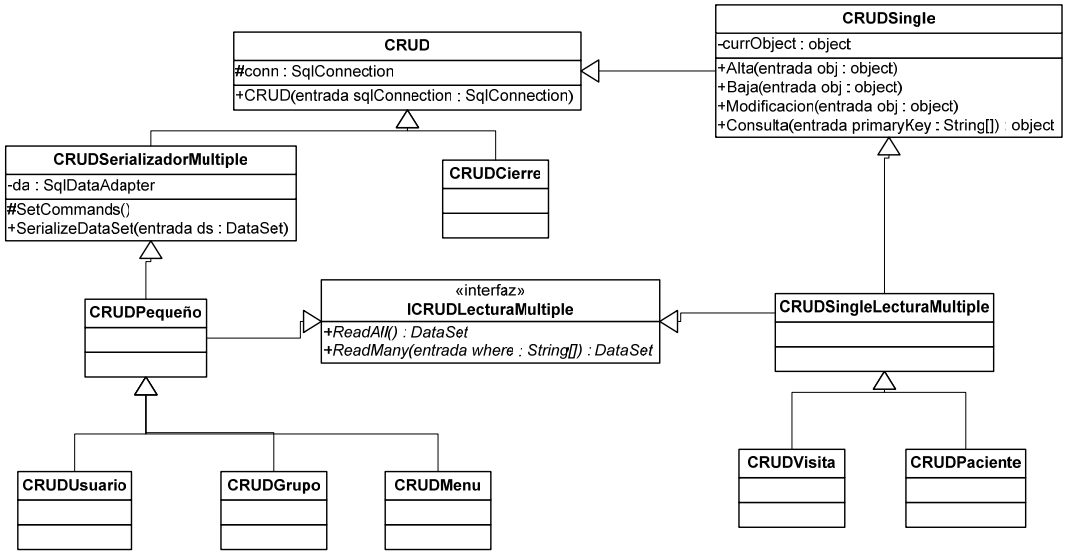
Las entidades aportarán también los métodos de clase necesarios.

No es necesario comentarlas ya que entre este diagrama y el diagrama ER podemos ver claramente cuales serán los campos necesarios en estas entidades.

5. 2.- Clases de acceso a datos

El diagrama de los CRUDS que puede verse a continuación representa todas aquellas clases que tendrán acceso a datos, en concreto a SQL Server.

Establecer una jerarquía de CRUDs es positivo para estandarizar sus métodos y su forma de trabajar.



En general con las entidades empresariales tenemos una serie de necesidades en función de la pantalla en la que nos encontremos, veámoslas:

1. Trato individual ya sea para guardar, modificar o guardar una entidad empresarial (currObject): Se refiere a que la entidad empresarial será tratada de una en una, por ejemplo la pantalla de Gestión de Pacientes da de alta o modifica un solo paciente. Por ello los CRUDs con trato individual heredan de CRUDSingle.

2. Presentación colectiva (Desserialización múltiple): Se refiere a que se necesitará presentar de la misma entidad diferentes instancias. Por ejemplo en la consulta de pacientes necesitamos visualizar varios simultáneamente. Por ello los CRUDs con presentación colectiva implementan ICRUDLecturaMultiple.

3. Serialización y borrado colectivo: Quiere decir que diversas instancias de una misma entidad empresarial podrán ser modificadas o insertadas o borradas, la operación se hará de forma colectiva, por ejemplo, las pantallas de gestión de usuarios, de grupos y de menús. Por ello los CRUDs con Serialización y borrado colectivo heredan de CRUDSerializadorMultiple

Además estas necesidades las podemos combinar, por ejemplo, el paciente es dado de alta y modificado en una pantalla pero cuando se muestra en una consulta es presentado junto con otros pacientes. Observamos que le entidad empresarial Paciente necesitará trato individual y presentación colectiva.

La jerarquía de CRUDS se ha diseñado para dar respuesta a todas estas necesidades.

CRUD

Clase base de todos los CRUDs. CRUD tiene una conexión SqlConnection ya que la aplicación trabaja con SQL Server. Es abstracta.

- Constructor CRUD (sqlConnection : SqlConnection)

Se encarga de crear el CRUD y proporcionarle la conexión de acceso a datos. Al pasárselo al CRUD se evita crear el acceso a datos continuamente.

CRUDSingle

Clase abstracta que representa los CRUDs con tratamiento individual.

La propiedad protegida currObject representa la última entidad devuelta o manipulada, puede ser útil por ejemplo para deshacer cambios.

Se puede ver que los métodos trabajan con object, se entiende que las subclases de CRUDSingle harán un casting a la entidad empresarial a la que corresponda. Por ejemplo desde CRUDUsuario haríamos (Usuario) currObj

- Alta (obj : Object)

Se encarga de dar de alta la entidad empresarial obj en SQL Server. Es virtual y abstracto.

- Baja (obj: Object).

Da de baja la entidad empresarial obj que se pasa como parámetro. Es virtual y abstracto.

- Modificación (obj : Object)

Modifica un a entidad empresarial obj que se pasa como parámetro. Es virtual y abstracto.

- Consulta (primaryKey : String[]) : Object

Devuelve la entidad empresarial que corresponde a los datos de la primary key que se pasan por parámetro. Es virtual y abstracto.

CRUDSerializadorMultiple

Clase abstracta que representa los CRUDs con serialización múltiple. Estos CRUDs se caracterizan por recibir una colección de entidades empresariales representadas mediante un DataSet, las operaciones que se llevan a cabo vienen establecidas en el DataSet, para ello se utiliza un DataAdapter que tendrá los comandos apropiados para cuando se haga Update.

- Propiedad DataAdapter da

DataAdapter que contendrá el InsertCommand y/o el UpdateCommand y/o el DeleteCommand apropiados

- SetCommands()

Establece los comandos al DataAdapter. Es virtual y abstracto.

- SerializeDataSet(ds: DataSet). Es virtual.

Lleva a cabo las operaciones con el Dataset que recibe por parámetro. La implementación en ésta clase será da.Update (ds) pero será virtual para que pueda redefinirse convenientemente si fuera necesario.

ICRUDLecturaMultiple

Interface que implementarán los CRUDs que necesitan presentar una colección de las entidades empresariales que manejan. Dicha colección será un DataSet.

- ReadAll() : DataSet

Método que devuelve absolutamente todas las entidades. Es muy peligroso en tablas grandes ya que devolver todos los registros es impensable y debe tenerse en cuenta. Es virtual y abstracto.

- ReadMany(where String[]) : DataSet

Recibe como parámetros las claves (el filtro) que devolverán los resultados, lo que en SQL formaría el where de la select. Es virtual y abstracto.

Por ejemplo, si habláramos de pacientes que viven en Barcelona podríamos recibir tan sólo “Barcelona”, CRUDxxx.ReadMany(new string[] {"Barcelona"}). ReadMany es muy genérica y siempre quedaría mejor por ejemplo CRUDxxx.GetPorCiudad(“Barcelona”), sería más legible, pero si el CRUD sólo tiene un tipo de consulta, ReadMany es suficiente.

CRUDSingleLecturaMultiple

Clase abstracta pensada para manejar entidades sobre las que se realizan operaciones individualmente pero que además necesitan ser consultadas de forma colectiva (Normalmente se buscan desde pantallas de consulta)

CRUDVisita

CRUD para el almacenar/obtener visitas. En concreto de las visitas se necesita:

- Manipular individualmente
- Obtener una
- Obtenerlas en base a unos criterios
- Marcarla con un estado concreto (como “Se ha presentado”, “Se realizó”...)
- Cambiar el estado de varias de “No realizada” a “No se realizó” de un determinado día, será necesario desde el proceso de cierre
- Borrar las anteriores a una determinada fecha

Las tres primeras operaciones las obtenemos redefiniendo las heredadas de CRUDSingleLecturaMultiple

- SetEstado (String Idivisita, char[1] estado)

Pone un nuevo estado a una visita. De momento estado es char[1] aunque se creará un enumerativo de estados, pero al guardarse en B.D. se guardará un char de uno.

- Cierre (DateTime diaCierre)

Cambia el estado de las visitas de diaCierre a “No se realizó” siempre y cuando estuvieran en “No realizada”.

- GetHistorial(String dniPaciente)

Devuelve las visitas del paciente que forman su historial

- EliminarAnteriores (DateTime fecha)

Elimina las visitas anteriores a fecha

CRUDPaciente

Crud para el almacenar/obtener pacientes. En concreto de los pacientes se necesita:

- Manipular individualmente
- Obtener uno
- Obtener varios en base a unos criterios
- Obtener el historial
- Hacer Login
- Obtener una lista de con o sin e-mail y que se visitarán dentro de x días.
- Eliminar los que no tengan visitas

Las tres primeras operaciones las obtenemos redefiniendo las heredadas de CRUDSingleLecturaMultiple

- Login (String Nif, String password) : bool

Método para hacer login de un paciente, si el login ha sido correcto devuelve true.

- ObtenerSeVisitaran (DateTime diaAVisitarse, bool soloConEmail) : DataSet

Obtiene los pacientes (que tienen email si soloConEmail es true) y que se visitarán en diaAVisitarse.

- EliminarPacientesSinVisitas()

Elimina los pacientes que no tengan visitas.

CRUDPequeño

Clase abstracta pensada para manejar entidades sobre las que se realizan operaciones colectivamente. Por ejemplo, en la aplicación a priori no será nunca necesario instanciar un menú, ni tan siquiera será necesario tratarlo

individualmente, no obstante se trabajará con colecciones de menús, las colecciones serán un DataSet.

Un CRUDPequeño es ideal para pantallas de gestión que tratan las entidades empresariales de forma colectiva y que realizan operaciones simultáneamente en distintas instancias.

Un CRUDPequeño especialmente tratará tablas con pocos registros y la implementación del método ReadAll de la interfície ICRUDLecturaMultiple no presentará mayor problema.

Un CRUDPequeño no es más que un CRUDSerializadorMúltiple que implementa la interface ICRUDLecturaMultiple.

CRUDUsuario

Crud para el almacenar/obtener usuarios. En concreto de los usuarios se necesita:

- Obtener todos los usuarios (como colección)
- Manipular usuarios de forma colectiva, como colección
- Hacer login de un usuario
- Comprobar si un usuario existe en la base de datos
- Cambiar la contraseña de un usuario (la del usuario activo en la aplicación)

Las dos primeras operaciones las obtenemos redefiniendo las heredadas de CRUDPequeño.

- Login (String idUsuario, String password) : Usuario

Método para hacer login de un usuario, si el login ha sido correcto devuelve el objeto Usuario

- Consulta (String idUsuario) : Usuario

Método que obtiene un usuario dado su id, si el usuario no existe devuelve null, se llamará desde dentro de Login

- SetPassword (String idUsuario, String antiguoPassword, String nuevoPassword)

Cambia la contraseña de un usuario

Debemos tener en cuenta que Usuario tendrá un DataSet con 2 DataTables, 1 para los grupos y otro para los menús que le son visibles.

CRUDGrupo

Crud para el almacenar/obtener grupos. En concreto de los grupos se necesita:

- Obtener todos los grupos (como colección)
- Manipular grupos de forma colectiva, como colección

Las operaciones las obtendremos redefiniendo las heredadas de CRUDPequeño.

Debemos tener en cuenta que Grupo tendrá un DataSet con 2 DataTables, 1 para los usuarios y otro para los menús que le son visibles.

CRUDMenu

Crud para el almacenar/obtener menús. En concreto de los menús se necesita:

- Obtener todos los menús (como colección)
- Manipular menús de forma colectiva, como colección

Las operaciones las obtendremos redefiniendo las heredadas de CRUDPequeño.

5. 3.- Clases de la vista o GUI

Puesto que las ventanas son clases y además debe planificarse su presentación separaremos este apartado en dos:

- Diagrama estático de clases
- Estructura de la interficie (Vista o GUI)

5. 3. 1. – Diagrama estático de clases

Observemos que existirán casos similares, por ejemplo:

- Los menús, usuarios y grupos son tablas pequeñas (con pocos registros) y los mantenimientos de las mismas se han pensado a modo de una rejilla en la que podremos llevar a cabo todas las operaciones.

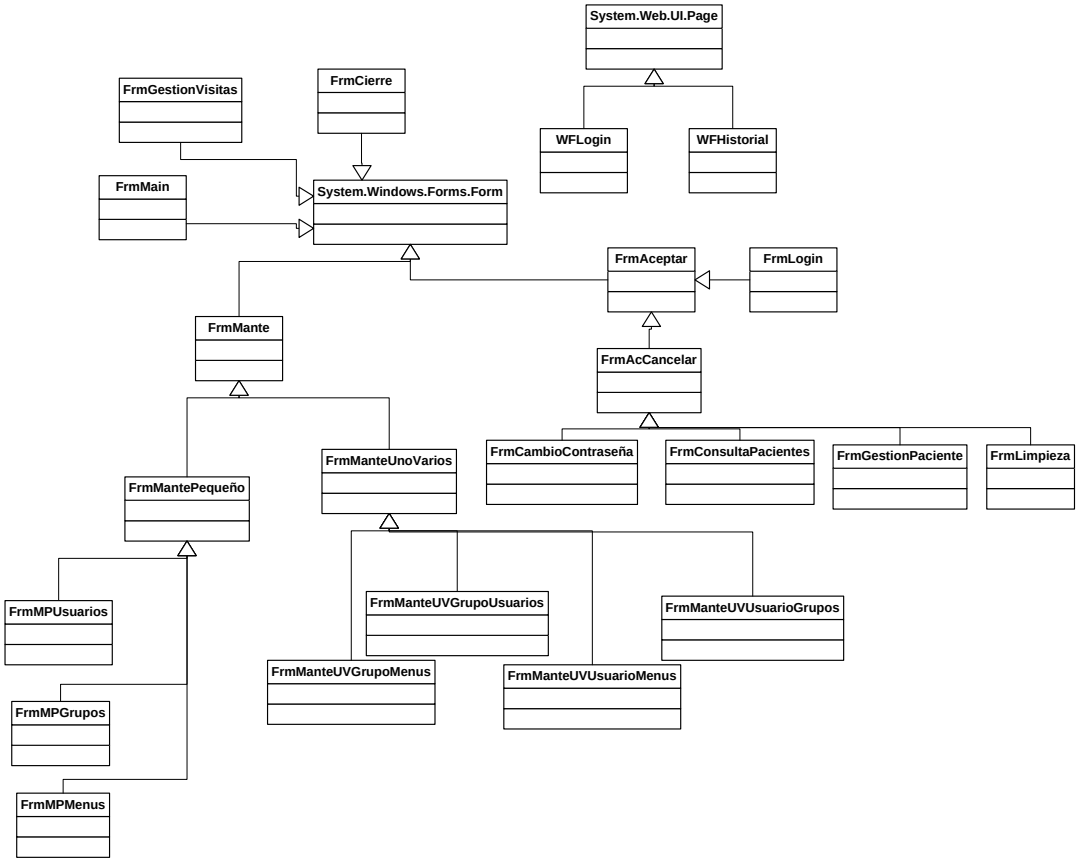
- Las relaciones usuario – menús, grupo – menús, grupo – usuarios, usuario – grupos podrían llevarse a cabo en pantallas muy similares.

- Además las pantallas que he comentando en los dos puntos anteriores son de hecho un mantenimiento, son diferentes pero no dejan de ser un mantenimiento.

- También hay pantallas en la aplicación que tienen similitudes en la forma en que son presentadas, por ejemplo muchas presentan dos botones (aceptar y cancelar)

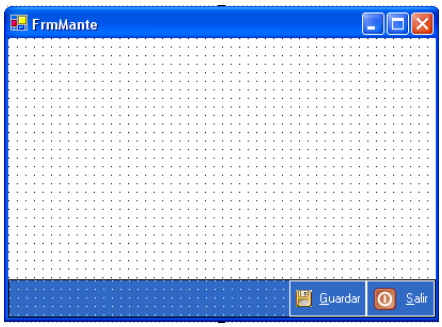
Puesto que estas pantallas tienen muchas similitudes, vemos que podríamos hablar de herencia entre pantallas ya que los controles pueden ser los mismos, incluso en algunos casos el funcionamiento.

Veamos la jerarquía de clases de los WinForms para el proyecto.



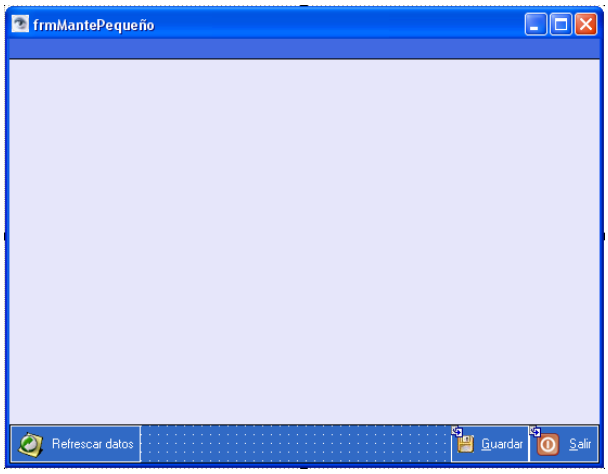
FrmMante

Clase padre de los mantenimientos. Se caracteriza porque presenta dos botones, guardar y salir. Definirá métodos virtuales y estandarizará el funcionamiento de los mantenimientos.



FrmMantePequeño

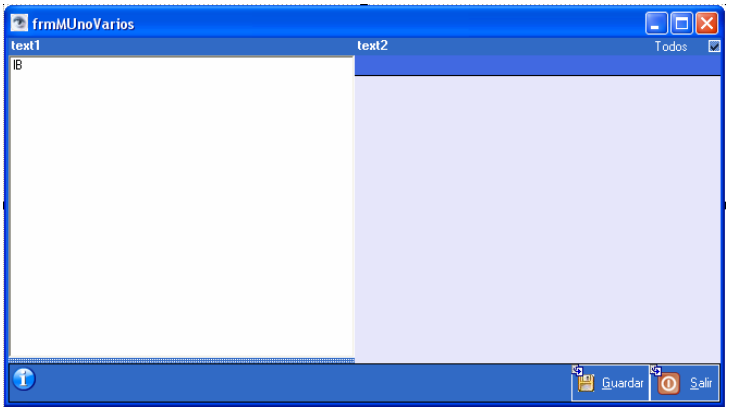
Clase padre de los mantenimientos que “mueven” tablas con pocos registros. Las subclases de GestorMPequeño manejarán estas pantallas.



Se caracterizan también porque el mantenimiento trata una colección de entidades, por lo que no se instancian y por tanto no necesita una entidad empresarial ya que trabajará directamente con un CRUD. Esta situación indica que hay Gestores que no mueven entidades empresariales y que sólo tienen una pantalla y un CRUD.

FrmManteUnoVarios

Clase padre de los mantenimientos sobre tablas fruto de una relación muchos a muchos.



Esta ventana presenta dos tablas, una a la izquierda que llamaremos principal y otra a la derecha en la rejilla que llamaremos secundaria.

El gestor de la ventana necesitará un CRUD para la principal y un CRUD para la secundaria, ambos deberán poder devolver todos los registros (entidades) de las tablas que mueven, por tanto implementarán ICRUDLecturaMultiple.

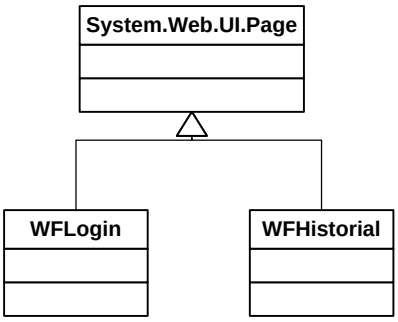
Al CRUD de la principal le deberemos poder pedir además la entidad seleccionada, una vez nos retorna la entidad seleccionada accederemos a su campo pertinente para obtener las entidades secundarias relacionadas. Después accederemos al CRUD de la secundaria para obtener todas las entidades y marcaremos las entidades secundarias relacionadas de la principal.

Cuando se pulse guardar se grabaran las entidades secundarias marcadas y/o desmarcadas según proceda para la entidad principal. Observemos que los cambios no se realizan ni sobre la tabla principal ni sobre la secundaria, se realizan en realidad en un tabla de relación entre ambas fruto de una relación muchos a

muchos, el CRUD responsable de aplicar los cambios a la BD podría ser el de la principal, el de la secundaria o se podría definir un CRUD para las tablas relacionadas. Será el CRUD de la principal el encargado de añadir las relaciones con la tabla secundaria de forma que el código podría quedar CRUDPrincipal.SetSecundarios(DataSet ds);

Si pensamos por ejemplo en Usuario – Grupos quedaría:
CRUDUsuario.SetGrupos (DataSet dsGrupos);

En la parte ASP .Net, tan sólo hay dos páginas:



Como se ha comentado anteriormente las entidades empresariales estarán totalmente desvinculadas del acceso a datos, además, los gestores son los encargados del buen funcionamiento de las pantallas.

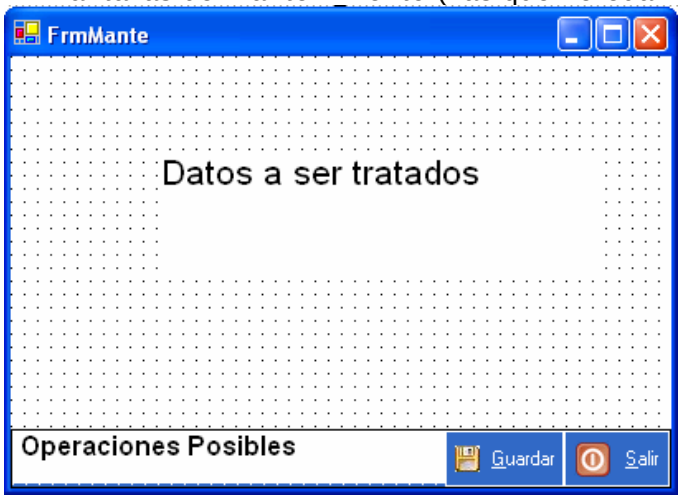
Los gestores pueden utilizar CRUDS, entidades empresariales y una pantalla.

5. 3. 2. – Estructura de la interfície (Vista o GUI)

Como en el resto del documento este apartado también se divide en dos partes, la arquitectura de las interficies de la parte de Winforms y de la parte de ASP .Net.

En la aplicación hablamos de diferentes tipos de ventanas.

- Pantallas de mantenimiento (Las que heredan de FrmMante)



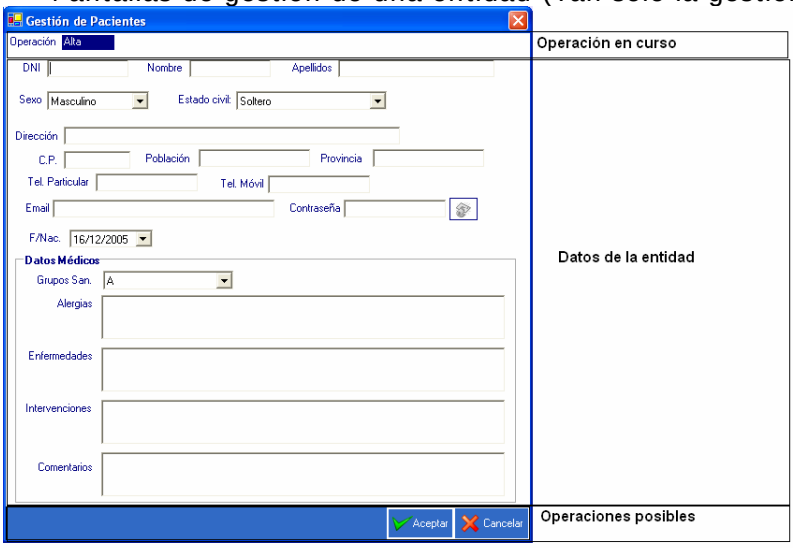
Las pantallas que van a ejercer de mantenimiento o gestión heredarán de FrmMante.

Añadirán controles en la zona “Datos a ser tratados” en función de la entidad/es que gestionen.

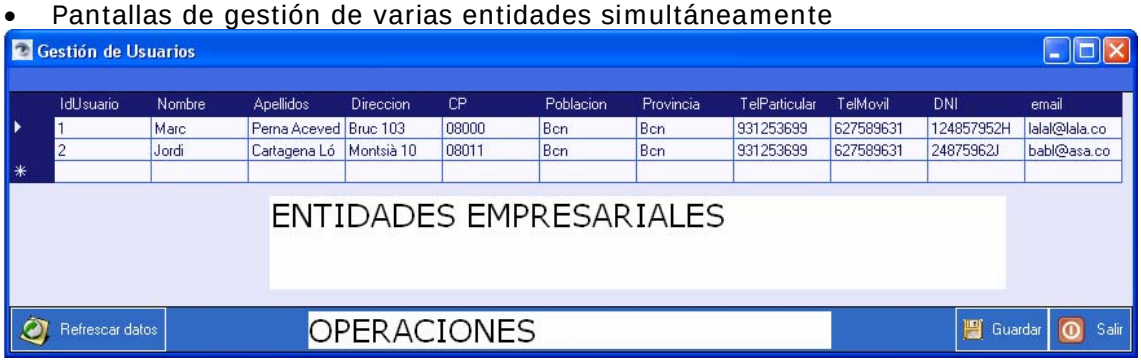
Las Operaciones básicas son guardar para aplicar los cambios y salir para cancelarlos si es necesario y salir, los formularios heredados podrán añadir sus propios botones.

Se estandariza de esta forma los mantenimientos de la aplicación.

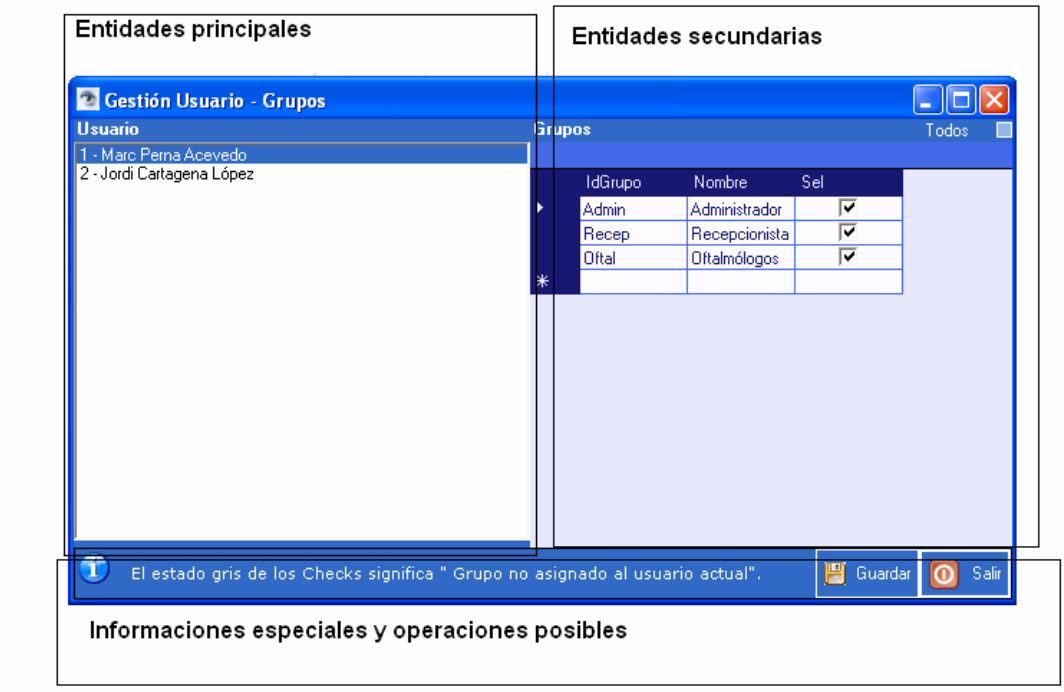
- Pantallas de gestión de una entidad (Tan sólo la gestión de pacientes)



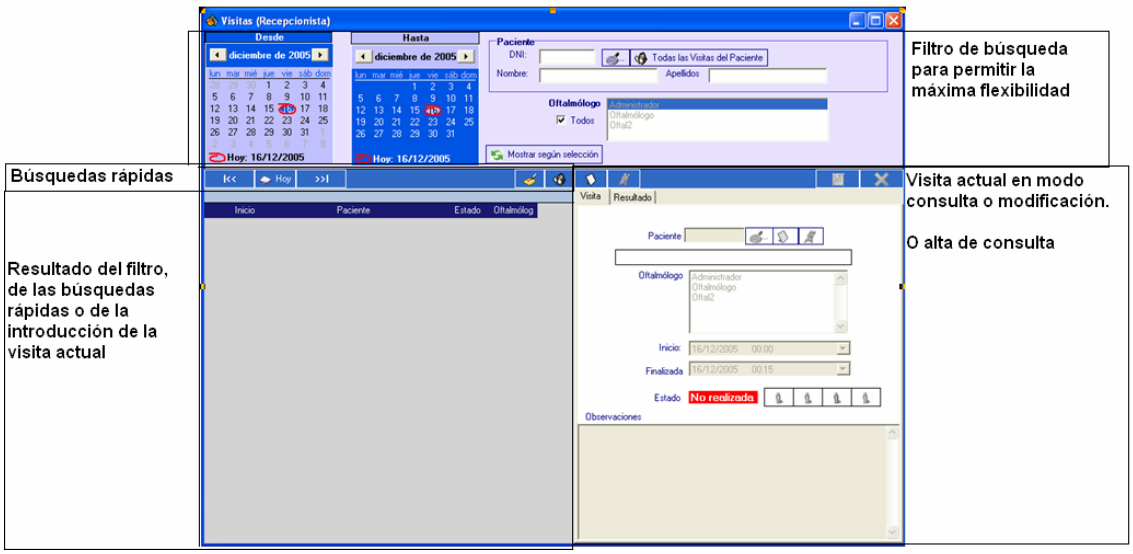
- Pantallas de gestión de varias entidades simultáneamente



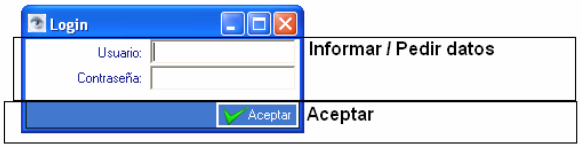
- Pantallas de gestión para relaciones muchos a muchos



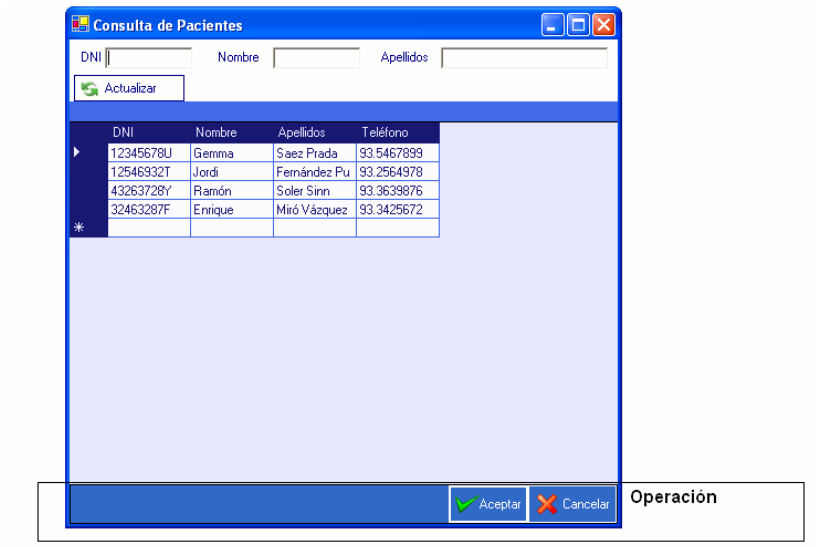
- Pantallas de gestión especial, Visitas



- Ventanas de sólo aceptación (Solo se puede aceptar o cerrar la ventana)



- Ventanas de sólo aceptar o cancelar.



Estas ventanas acostumbran a ser modales.

Toda la pantalla se adapta al cambio de tamaño de la ventana, los botones de operación siempre permanecen en la parte inferior derecha y la rejilla aumenta/disminuye su tamaño de forma automática.

5. 4.- Clases gestoras o del controlador

Hemos comentado que los gestores son los encargados del buen funcionamiento de las pantallas, las entidades empresariales y el acceso a datos, y es su responsabilidad coordinarlos.

También se ha comentado que se empleará el modelo MVC, por tanto cada pantalla tendrá un gestor y como mínimo habrá tantos gestores como pantallas.

Pensando en pantallas podemos afirmar que la aplicación tiene estos tipos de pantallas:

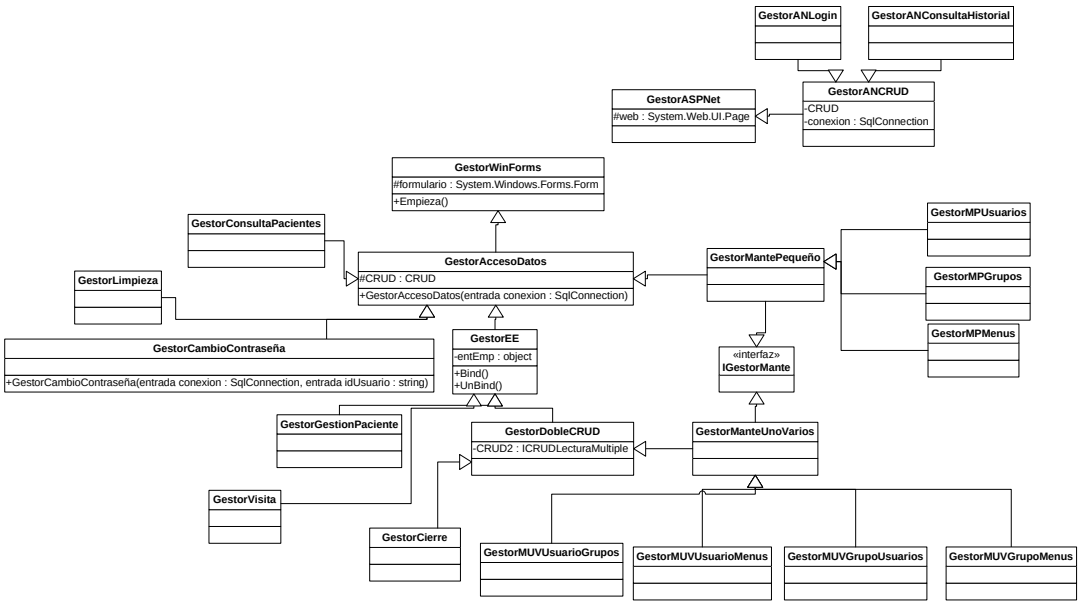
- Pantallas que presentan una sola entidad empresarial: Por ejemplo la pantalla de modificación/alta de un paciente.
- Pantallas que presentan una colección de entidades empresariales: Por ejemplo las pantallas FrmMantePequeño (Usuarios, Grupos,…)
- Pantallas que presentan dos colecciones de entidades empresariales diferentes: Por ejemplo las pantallas FrmManteUnoVarios necesita acceso a una entidad empresarial (la principal) y a dos CRUDs (el de la principal y el de la secundaria)
- Pantallas sin acceso a entidad empresarial: Por ejemplo la pantalla de cierre no presenta una entidad empresarial, ni tan siquiera la pide. La pantalla pide una serie de parametrizaciones para un proceso que se llevará a cabo, el de cierre.

Vistos los tipos de pantallas que los gestores deberán coordinar puede verse la necesidad de hablar de diferentes tipos de gestores:

- Gestores que necesitan presentar una entidad empresarial en pantalla y guardarla en la base de datos. Necesitan por tanto una entidad empresarial y un CRUD.
- Gestores sin entidad empresarial: Ya que nunca accederán individualmente a una entidad empresarial, siempre trabajarán con una colección de ellas. Necesitan por tanto un CRUD.
- Gestores con una entidad empresarial y dos CRUDs. Las pantallas FrmManteUnoVarios.
- Gestores que no presentan una entidad empresarial en pantalla y que su misión puede ser ejecutar procesos.

Podría haber otro tipo de pantalla que presentara una entidad empresarial y que permitiera o no variarla pero que no necesitara ser guardada en la base de datos y por tanto sólo tuviera efectos sobre la instancia. No obstante no he encontrado ninguna pantalla que deba funcionar así, de forma que se puede asegurar que todo gestor con entidad empresarial precisa de acceso a datos, es decir, que tendrá un CRUD.

Veamos el diagrama de clases como quedaría.



En primer lugar vemos una clara separación de gestores, los que parten de GestorWinForms y los que parten de GestorASPNet, podríamos haber empezado por un padre común para ambas ramas llamado Gestor, no obstante no nos serviría de nada ni tendría ninguna utilidad excepto la de ver claramente en el diagrama que ambas ramas son gestores pero por el nombre de las clases eso ya está claro.

GestorWinForms

Clase padre de todos los Gestores que asociaremos a un WinForm. Aporta el correspondiente campo que corresponderá al form asociado.

- Empieza()

Empezará la ejecución del Gestor, presentando el form asociado.

GestorAccesoDatos

Especialización de GestorWinForms que aporta lo necesario para poder manejar un objeto de acceso a datos, un CRUD.

- GestorAccesoDatos (SqlConnection conexión);

Constructor que recibe como parámetro la conexión activa de la aplicación. Puesto que Oftalmólogo 1.0 tan sólo trabajará con SQL Server recibe un SqlConnection. Si quisiéramos dejar la estructura de los Gestores más abierta pensando en que pudieran tener otros CRUDs que accedieran por ejemplo a MySql entonces podríamos pensar que el parámetro fuera IDbConnection, en cualquier caso queda fuera del alcance de este proyecto.

GestorEE

Especialización de GestorAccesoDatos que aporta lo necesario para poder manejar un objeto entidad empresarial.

- Bind ();

Establece el bind entre los controles del form y la entidad empresarial

- UnBind ();

Quita el bind entre los controles del form y la entidad empresarial.

GestorDobleCRUD

Especialización de GestorEE que aporta lo necesario para poder manejar otro CRUD más. Es una clase muy pensada para trabajar con FrmManteUnoVarios. No obstante podría ser de utilidad en un futuro para gestionar otras pantallas y por dejar la estructura lo más abierta posible, los FrmManteUnoVarios que creemos heredarán de una clase intermedia que implementará IGestorMante.

GestorConsultaPaciente

Gestionará FrmConsultaPacientes y puesto que sólo presentará una colección de pacientes lo único que necesitará es un CRUD.

GestorLimpieza

Gestionará FrmLimpieza

Necesita llamar al CRUD de la Visita para eliminar las visitas anteriores a una fecha y al CRUD de los pacientes para eliminar los pacientes sin visita.

GestorCambioContraseña

Gestionará FrmCambioContraseña.

- GestorCambioContraseña (SqlConnection conexión, string idUsuario);

Recibe por parámetro el id del usuario de la aplicación, necesita llamar al CRUD de usuarios para pasarle los datos introducidos en la pantalla y así cambiar la contraseña del usuario.

GestorGestionPaciente

Gestionará FrmGestiónPaciente.

Necesita una entidad empresarial Paciente y llamar al CRUD de pacientes para dar de alta, consultar o modificar la entidad empresarial.

GestorCierre

Encargado de realizar el cierre diario. Trabaja con dos CRUDS el de Visitas para cambiar el estado de las visitas en el día del cierre y el CRUD de los Pacientes para obtener aquellos que se visitan en una determinada fecha y que tienen e-mail o no.

GestorMantePequeño

Gestionará FrmMantePequeño.

Definirá una serie de métodos (algunos virtuales) y comportamientos pensados para gestionar un FrmMantePequeño y un CRUDPequeño, además puesto que gestionará un FrmMante implementa IGestorMante.

GestorMPUsuarios

Gestionará FrmMPUsuarios.

Redefinirá los métodos apropiados de GestorMantePequeño y utilizará CRUDUsuario.

GestorMPGrupos

Gestionará FrmMPGrupos.

Redefinirá los métodos apropiados de GestorMantePequeño y utilizará CRUDGrupo.

GestorMPMenu

Gestionará FrmMPMenu.

Redefinirá los métodos apropiados de GestorMantePequeño y utilizará CRUDMenu.

GestorManteUnoVarios

Gestionará FrmManteUnoVarios.

Definirá una serie de métodos (algunos virtuales) y comportamientos pensados para gestionar un FrmManteUnoVarios, dos ICRUDLecturaMultiple y una entidad empresarial, además puesto que gestionará un FrmMante implementa IGestorMante.

GestorManteUVUsuarioGrupos

Gestionará FrmManteUVUsuarioGrupos.

Redefinirá los métodos apropiados de GestorManteUnoVarios y utilizará los CRUDs CRUDUsuario y CRUDGrupo. También utilizará la entidad empresarial Usuario.

GestorManteUVUsuarioMenus

Gestionará FrmManteUVUsuarioMenus.

Redefinirá los métodos apropiados de GestorManteUnoVarios y utilizará los CRUDs CRUDUsuario y CRUDMenu. También utilizará la entidad empresarial Usuario.

GestorManteUVGrupoUsuarios

Gestionará FrmManteUVGrupoUsuarios.

Redefinirá los métodos apropiados de GestorManteUnoVarios y utilizará los CRUDs CRUDGrupo y CRUDUsuario. También utilizará la entidad empresarial Grupo.

GestorManteUVGrupoMenus

Gestionará FrmManteUVGrupoMenus.

Redefinirá los métodos apropiados de GestorManteUnoVarios y utilizará los CRUDs CRUDGrupo y CRUDMenus. También utilizará la entidad empresarial Grupo.

GestorASPNet

Clase padre de todos los Gestores que asociaremos a una Page. Aporta el correspondiente campo que corresponderá a la Page asociada.

- Empieza()

Empezará la ejecución del Gestor

GestorANCrud

Especialización de GestorAspNet que aporta lo necesario para poder manejar un objeto de acceso a datos, un CRUD.

GestorLogin

Gestionará la entrada del usuario al entorno Web. Utilizará CRUDPaciente para validar al paciente, si lo valida correctamente guardará su Nif en la sesión.

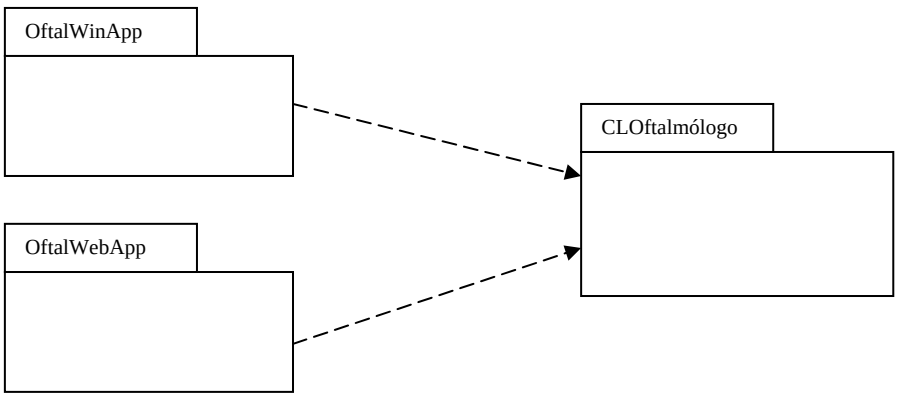
GestorHistorial

Mostrará el historial de un paciente. Para ello recogerá el Nif de la sesión y utilizará CRUDPaciente para consultar su historial.

5. 5.- Paquetes

La división lógica de la aplicación será en 3 paquetes:

- CLOftalmólogo: Class Library que incluirá,
  - Entidades empresariales, las entidades empresariales son comunes al cliente Windows y a la aplicación Web
  - CRUDS, el acceso a datos será el mismo tanto para la parte Windows como para la parte Web
  - Clases como excepciones, utilidades … que podrán ser utilizadas tanto por la parte Windows como por la parte Web
- OftalWinApp: Paquete de la aplicación Windows que incluirá,
  - Gestores (Árbol GestorWinForms)
  - Interficies de usuario
- OftalWebApp: Paquete de la aplicación Web que incluirá,
  - Gestores (Árbol GestorANCRUD)
  - Interficies de usuario

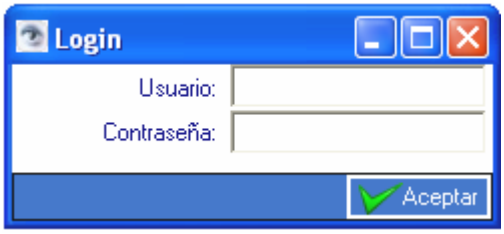


6.- Descripción de Oftalmólogo 1.0

A continuación describiré las funcionalidades más representativas de la aplicación, comentando la interfície gráfica y explicando las operativas que se pueden llevar a cabo.

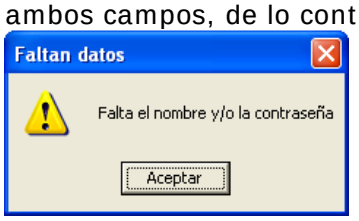
6. 1.- Subsistema de conexión

Pantalla de conexión (o Login)



La pantalla es llamada desde el Main del proyecto.

Se presenta la pantalla solicitando el usuario y contraseña, es obligatorio introducir ambos campos, de lo contrario no dejará continuar y mostrará el mensaje:



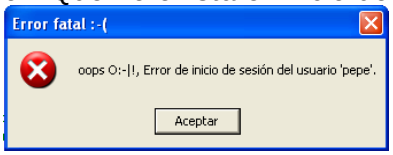
Si la aplicación ha sido ejecutada por accidente es posible cerrar por el botón de la ventana y no se mostrará ningún mensaje.

Una vez introducido el usuario y contraseña se lee el fichero Config.XML y se extrae la cadena de conexión de la etiqueta DBConnection, posteriormente se formatea con el usuario y contraseña introducidos y se procede a intentar el inicio de sesión de SQL Server.

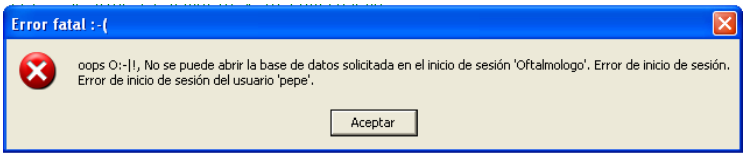
En caso de que el usuario y contraseña sean correctos aparece la pantalla principal con los menús que puede visualizar, recordemos que los menús serán decididos en primera instancia por los menús asociados al usuario y posteriormente en base a los grupos a los que pertenezca.

Si por el contrario ha ido mal pueden suceder tres cosas:

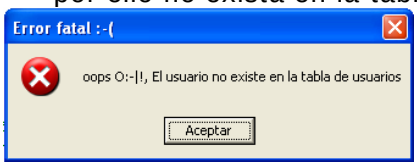
- Que no exista el inicio de sesión con lo que aparecerá el mensaje:



- Que exista el inicio de sesión y que no esté asignado a la base de datos



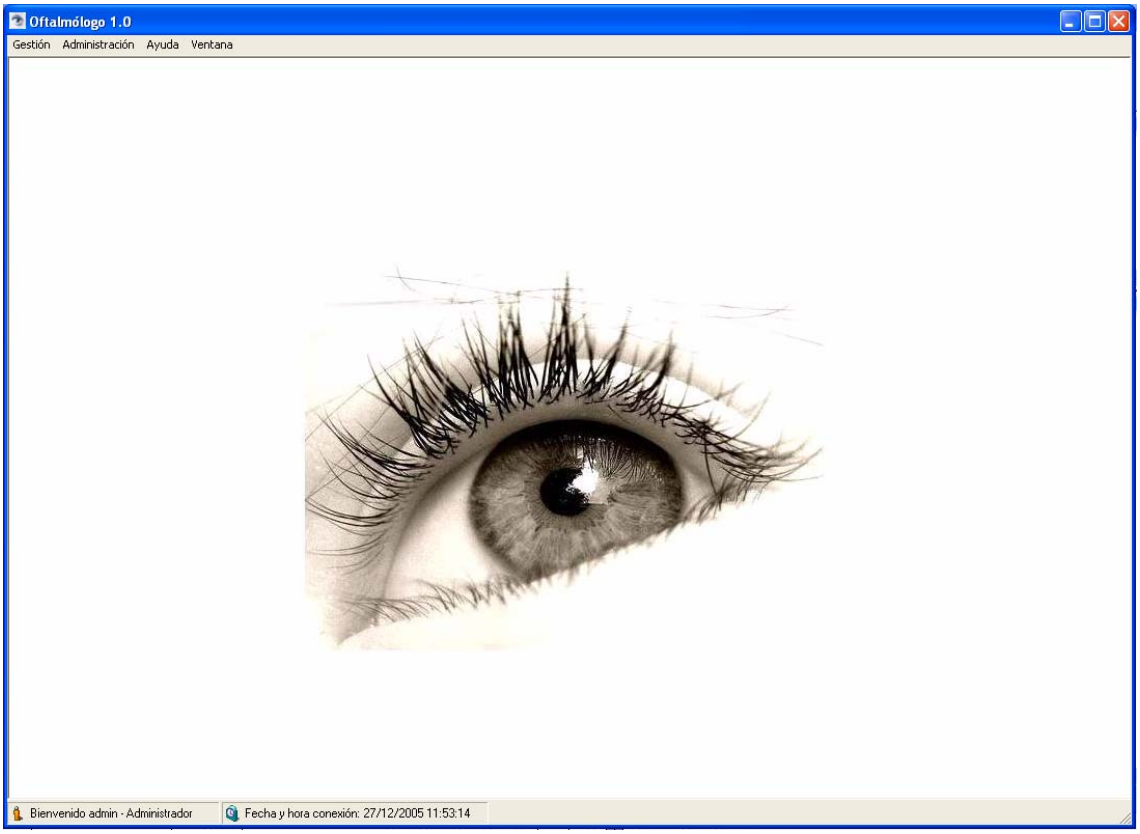
- Que externamente al programa se haya creado el inicio de sesión y concedido acceso a la base de datos con los permisos pertinentes y que por ello no exista en la tabla Usuario



Por defecto el script DBGeneracion.SQL crea 3 usuarios:

Usuario	Contraseña
Admin	Admin
Oftal	Oftal
Recep	recep

Pantalla principal



Esta pantalla muestra los menús accesibles para el usuario actual, los menús visibles se establecen en base a los permisos que tengan los grupos a los que pertenezca el usuario actual y los permisos asignados a nivel de usuario en última instancia.

La pantalla se muestra después de haber sido validado el usuario de entrada, muestra información del usuario actual y de la hora de conexión en la parte inferior.

Se trata de una pantalla MDI (Multiple Document Interface) y alojará todas las ventanas que sean llamadas desde los respectivos menús, eso sí, controla que una ventana no pueda ser instanciada dos veces.

6. 2.- Subsistema de seguridad o de administración

Usuarios

Desde esta pantalla llevaremos a cabo la gestión de usuarios de la aplicación. Por defecto oftalmólogo incorpora tres usuarios, admin, recep y oftal.

Funcionamiento:

Alta
Para dar de alta un usuario crearemos una nueva fila e introduciremos los datos. Si falta algún dato o se ha introducido erróneamente la aplicación nos avisará.

Cuando se crea un usuario nuevo automáticamente se crea un inicio de sesión en SQL Server concediéndole acceso a la base de datos de la aplicación

Modificación
Para modificar un usuario que acabamos de introducir o que ya existía nos situaremos en la fila del usuario y celda apropiada y modificaremos los datos.

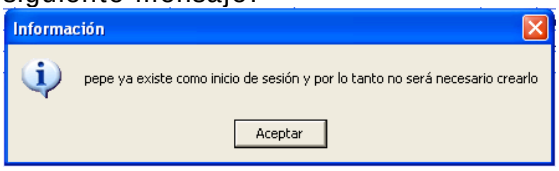
Baja
Para eliminar marcaremos la fila del usuario (pulsando la parte azul fuerte izquierda de la fila) y pulsaremos el botón supr.

A tener en cuenta
Los cambios no se aplican hasta que se pulsa el botón Guardar, si nos olvidamos de pulsarlo y salimos, la aplicación nos preguntará si deseamos guardar los cambios.

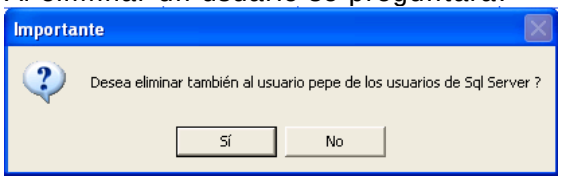
La pantalla realiza una serie de comprobaciones:

- Comprueba que los datos mínimos estén introducidos para permitir guardar el usuario
- Prohíbe cambiar el Id de un usuario a no ser que todavía no se haya guardado
- Prohíbe eliminar el usuario admin
- Prohíbe eliminar el usuario actual, es decir, si hemos entrado con Laura no dejará que eliminemos el usuario Laura

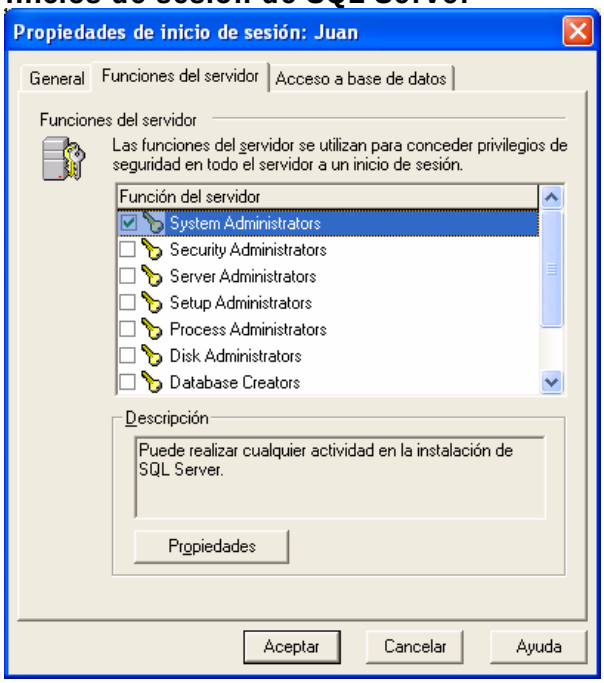
Si se crea un usuario y su inicio de sesión ya existía en SQL Server aparecerá el siguiente mensaje:



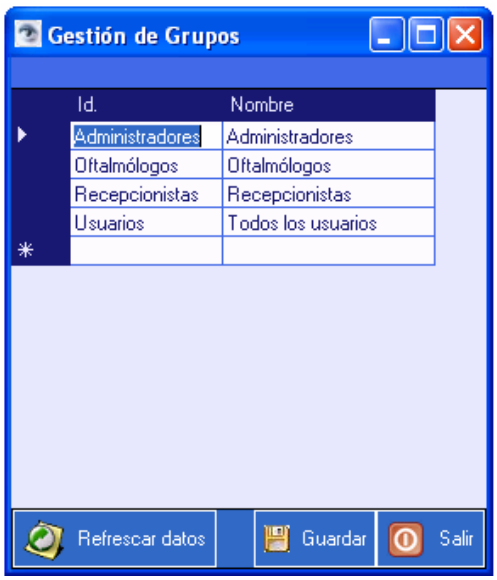
Al eliminar un usuario se preguntará:



*****Si se desea crear un usuario administrador, aparte de crearlo y añadirlo al grupo Administradores, será necesario concederle permisos de sysadmin en el administrador corporativo, ya que deberá poder eliminar usuarios o añadir nuevos y recordemos que estas funcionalidades alteran inicios de sesión de SQL Server**



Grupos de usuarios



Desde esta pantalla llevaremos a cabo la gestión de grupos de usuarios de la aplicación. Por defecto oftalmólogo incorpora cuatro grupos, Administradores, Oftalmólogos, Recepcionistas y Usuarios

Funcionamiento:

Alta
Para dar de alta un grupo crearemos una nueva fila e introduciremos los datos. Si falta algún dato o se ha introducido erróneamente la aplicación nos avisará.

Modificación
Para modificar un grupo que acabamos de introducir o que ya existía nos los situaremos en la fila del grupo y celda apropiada y modificaremos datos.

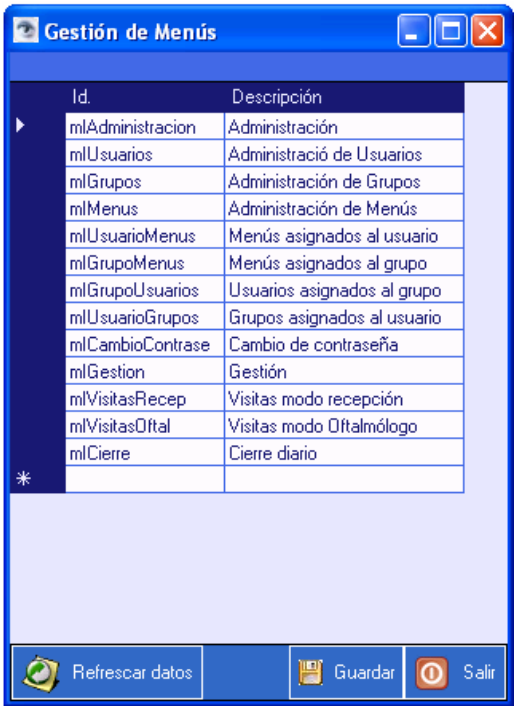
Baja
Para eliminar marcaremos la fila del grupo (pulsando la parte azul fuerte izquierda de la fila) y pulsaremos el botón supr.

A tener en cuenta
Los cambios no se aplican hasta que se pulsa el botón Guardar, si nos olvidamos de pulsarlo y salimos, la aplicación nos preguntará si deseamos guardar los cambios.

La pantalla realiza una serie de comprobaciones:

- Comprueba que los datos mínimos estén introducidos para permitir guardar el grupo
- Prohíbe cambiar el Id de un grupo a no ser que todavía no se haya guardado
- Prohíbe eliminar los grupos Administradores y Oftalmólogos.

Menús



Desde esta pantalla llevaremos a cabo la gestión de menús de la aplicación. No olvidemos que la pantalla ha sido diseñada para el desarrollador, de forma que fácilmente pueda añadir menús en la base de datos para poder asignarlos a su usuario y/o grupo para visualizarlos, por lo tanto y como he comentado anteriormente es una pantalla de desarrollo que ni tan siquiera el administrador en condiciones normales debería ver, pero que se incluye ya que ha sido desarrollada y forma parte del proyecto.

Funcionamiento:

Alta
Para dar de alta un menú crearemos una nueva fila e introduciremos los datos.
Si falta algún dato o se ha introducido erróneamente la aplicación nos avisará.

Modificación
Para modificar un menú que acabamos de introducir o que ya existía nos situaremos en la fila del grupo y celda apropiada y modificaremos los datos.

Baja
Para eliminar marcaremos la fila del menú (pulsando la parte azul fuerte izquierda de la fila) y pulsaremos el botón supr.

A tener en cuenta
Los cambios no se aplican hasta que se pulsa el botón Guardar, si nos olvidamos de pulsarlo y salimos, la aplicación nos preguntará si deseamos guardar los cambios.

La pantalla realiza una serie de comprobaciones:
- Comprueba que los datos mínimos estén introducidos para permitir guardar el grupo
- Prohíbe cambiar el Id de un menú a no ser que todavía no se haya guardado

Grupo - Usuarios

Desde esta pantalla podremos establecer los usuarios que son miembros de un determinado grupo.

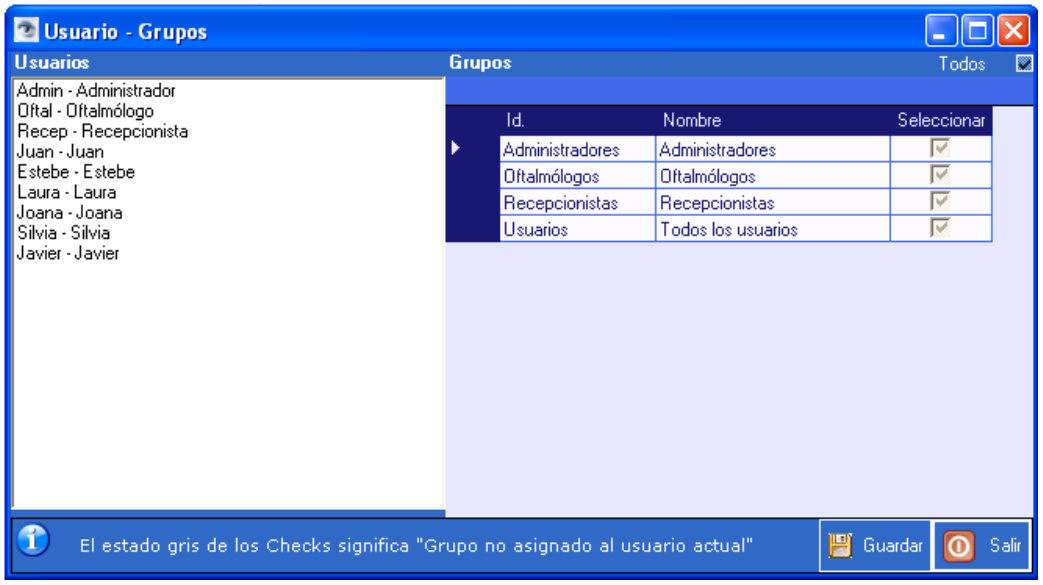
Operativa
El funcionamiento es muy simple, marcamos el grupo y vemos como en la parte de la rejilla aparecen marcados los usuarios que ya son miembros del grupo, si deseamos incluir nuevos usuarios los marcamos, en cambio, si deseamos eliminarlos del grupo los desmarcamos.

A tener en cuenta
Los cambios no se aplican hasta que se pulsa el botón Guardar, si nos olvidamos de pulsarlo y salimos, la aplicación nos preguntará si deseamos guardar los cambios.

Si hemos realizado cambios y cambiamos de grupo, la aplicación nos preguntará si deseamos guardar los cambios antes de cambiar de grupo, caso afirmativo se guardarán, caso negativo se perderán.

No se permite quitar el usuario Admin del grupo Administradores

Usuario - Grupos



Desde esta pantalla podremos establecer los grupos de los que es miembro un determinado usuario.

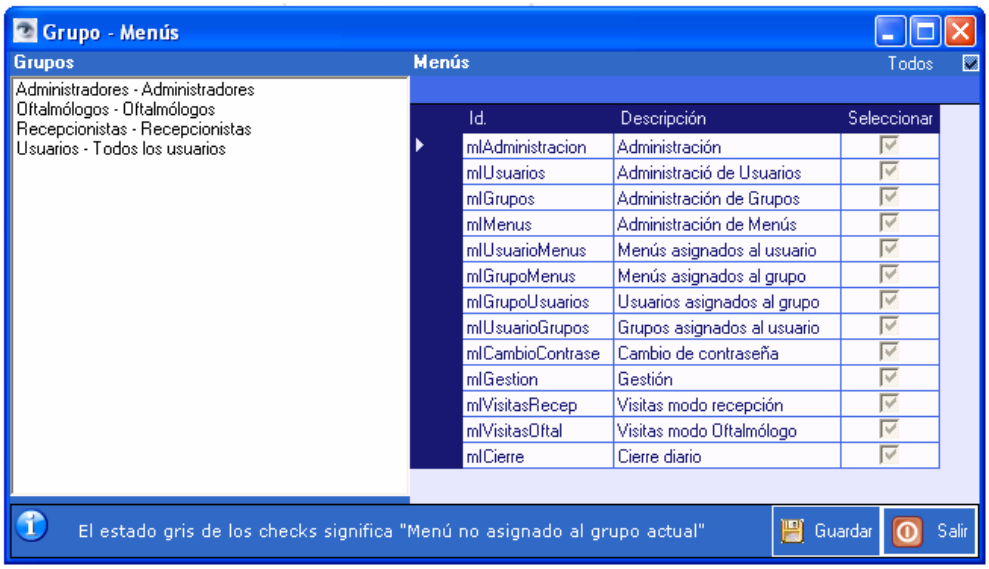
Operativa
El funcionamiento es muy simple, marcamos el usuario y vemos como en la parte de la rejilla aparecen marcados los grupos de los que es miembro, si deseamos incluirlo como miembro de nuevos grupos los marcamos, en cambio, si deseamos eliminar su pertenencia lo desmarcamos.

A tener en cuenta
Los cambios no se aplican hasta que se pulsa el botón Guardar, si nos olvidamos de pulsarlo y salimos por el botón Salir la aplicación nos preguntará si deseamos guardar los cambios.

Si hemos realizado cambios y cambiamos de usuario, la aplicación nos preguntará si deseamos guardar los cambios antes de cambiar de usuario, caso afirmativo se guardarán, caso negativo se perderán.

No se permite eliminar Administradores como grupo de Admin.

Grupo - Menús



Desde esta pantalla podremos establecer los menús que podrá visualizar un determinado grupo.

Operativa

El funcionamiento es muy simple, marcamos el grupo y vemos como en la parte de la rejilla aparecen marcados los menús visibles para el grupo, si deseamos incluir nuevos menús los marcamos, en cambio, si deseamos que no sean visibles para el grupo los desmarcamos.

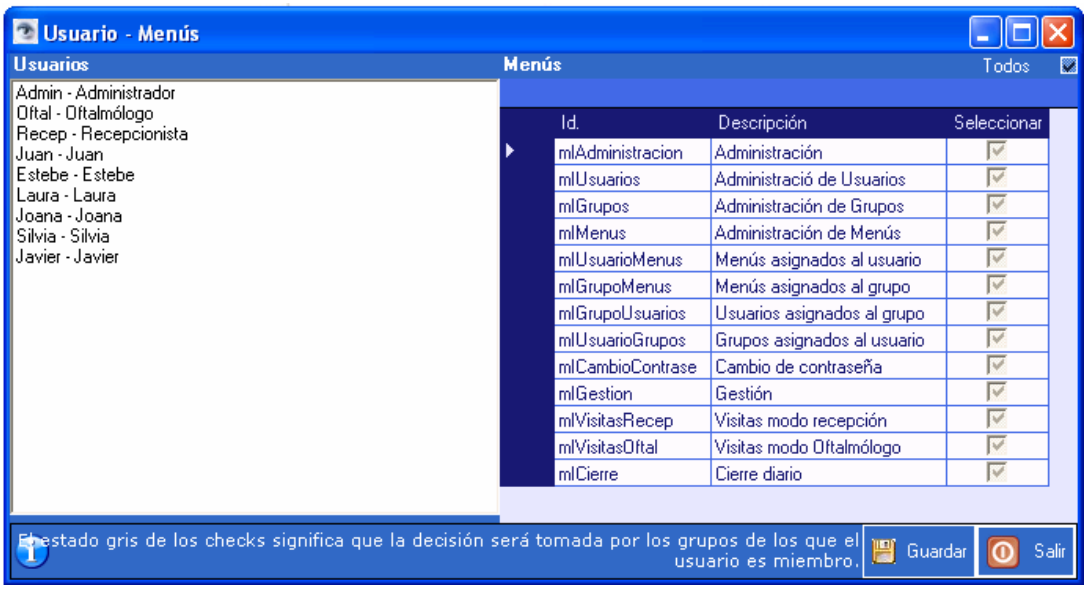
A tener en cuenta

Los cambios no se aplican hasta que se pulsa el botón Guardar, si nos olvidamos de pulsarlo y salimos, la aplicación nos preguntará si deseamos guardar los cambios.

Si hemos realizado cambios y cambiamos de grupo, la aplicación nos preguntará si deseamos guardar los cambios antes de cambiar de grupo, caso afirmativo se guardarán, caso negativo se perderán.

No se permite variar los menús del grupo Administradores

Usuario - Menús



Desde esta pantalla podremos establecer los menús que podrá visualizar un determinado usuario.

Operativa

El funcionamiento es muy simple, marcamos el usuario y vemos como en la parte de la rejilla aparecen marcados los menús visibles para el usuario, si deseamos incluir nuevos menús los marcamos, en cambio, si deseamos que no sean visibles para el usuario los desmarcamos.

A tener en cuenta

Los cambios no se aplican hasta que se pulsa el botón Guardar, si nos olvidamos de pulsarlo y salimos, la aplicación nos preguntará si deseamos guardar los cambios.

Si hemos realizado cambios y cambiamos de usuario, la aplicación nos preguntará si deseamos guardar los cambios antes de cambiar de grupo, caso afirmativo se guardarán, caso negativo se perderán.

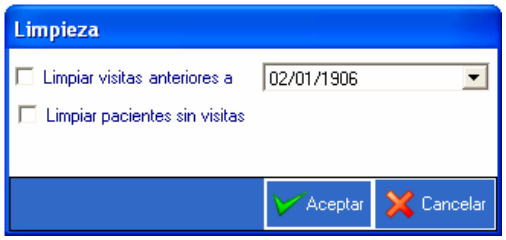
El estado gris de los checks será considerado como “la decisión de que el usuario pueda ver o no el menú será un OR de los permisos asignados a los grupos de los que sea miembro”.

El estado desmarcado de los checks será considerado como “el usuario no podrá ver el menú independientemente de lo establecido en los grupos de los que sea miembro”.

El estado marcado de los checks será considerado como “el usuario podrá ver el menú independientemente de lo establecido en los grupos de los que sea miembro”

No se pueden variar los menús visibles para el usuario Admin...

Limpieza

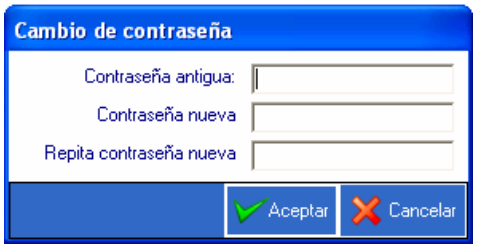


Desde esta pantalla podremos eliminar las visitas antiguas y los pacientes que hayan quedado sin visitas.

Operativa

El funcionamiento es muy simple, marcamos las operaciones que deseemos llevar a cabo y en caso de limpiar visitas introducir una fecha, pulsamos aceptar y se efectúa la operación.

Cambio de contraseña



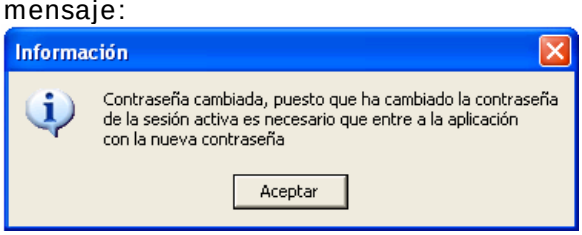
Desde esta pantalla podremos cambiar la contraseña del usuario actual, es decir, del usuario que ha iniciado el programa.

Operativa

El funcionamiento es muy simple, introducimos la contraseña actual y la nueva dos veces para confirmar que es válida, a continuación aceptamos y se realiza el cambio de contraseña.

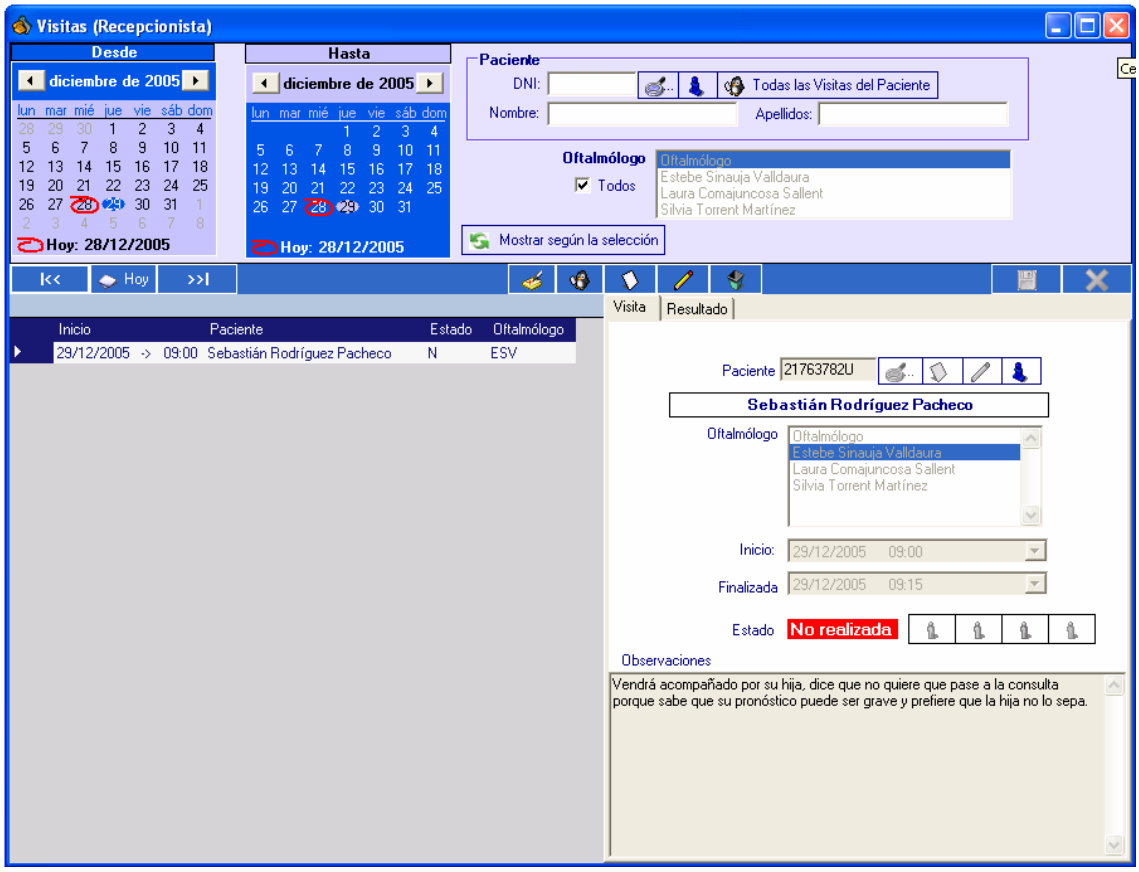
A tener en cuenta

El cambio de contraseña afecta al inicio de sesión de SQL Server actual, por tanto si cambiamos la contraseña la sesión actual cambia y se debe volver a establecer la conexión con la nueva contraseña, por ello la aplicación muestra el siguiente mensaje:



6. 3.- Subsistema de gestión

Visitas



La pantalla de visitas ofrece al usuario del centro oftalmológico el 99% de todas las acciones que lleva a cabo a lo largo del día. Se trata de una pantalla que al mismo tiempo es consulta y gestión, es decir, que permite consultar visitas en base a muchos criterios, añadir nuevas, modificarlas…

Una de las ventajas de permitir múltiples funcionalidades en la misma pantalla es que el usuario no necesitará navegar a través de ventanas ni tener varias abiertas simultáneamente, sin duda la complejidad de la programación aumenta por tener que controlar muchas más situaciones y combinaciones pero de cara al usuario las ventajas son muchas.

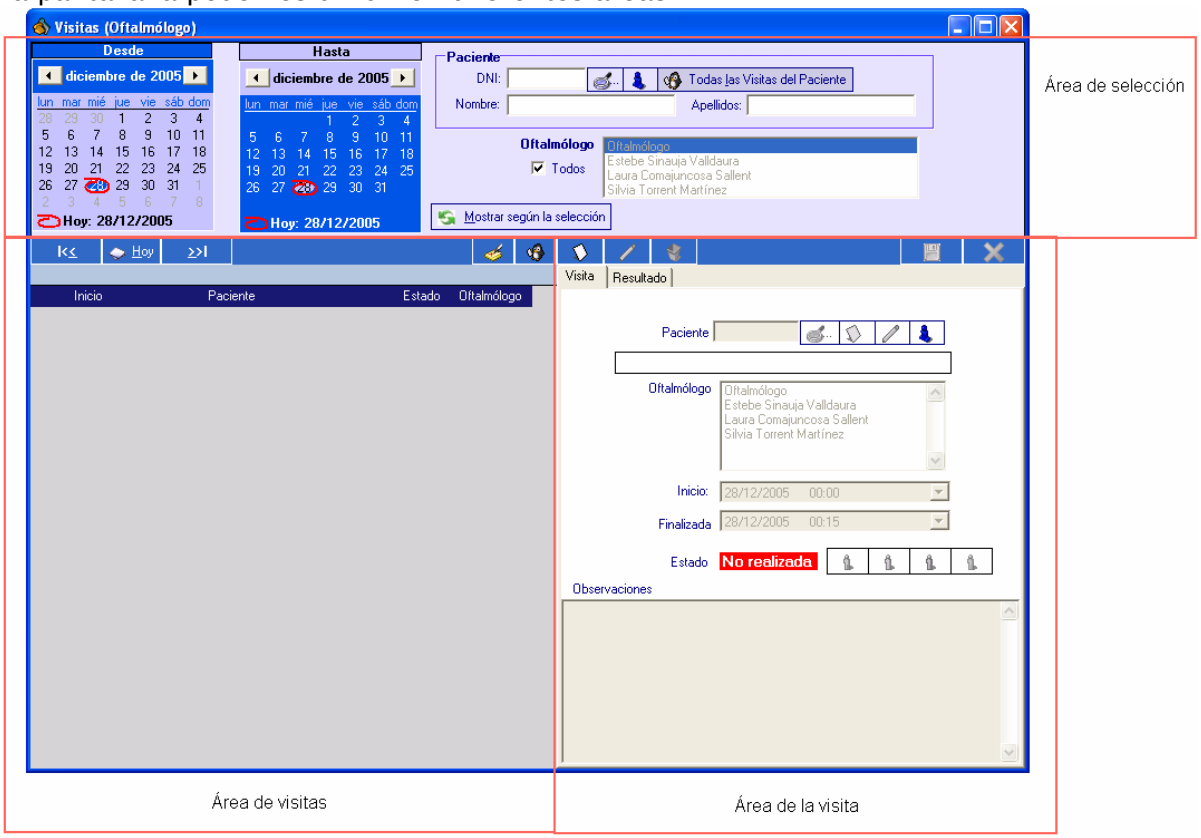
Algunas de las operaciones que se pueden llevar a cabo son:

- Consultar visitas en base a múltiples condiciones
- Consultar el historial de un paciente
- Consultar las visitas día a día
- Ver la ficha de un paciente
- Dar de alta un paciente
- Modificar un paciente
- Modificar una visita
- Introducir una nueva visita
- Eliminar una vista

Esta pantalla presenta más o menos datos e incluso cambia su comportamiento en función de si el usuario actual es un:

- Oftalmólogo: La pantalla aparecerá por defecto mostrando sólo las visitas asignadas al usuario actual. Cuando realice un alta, por defecto la nueva visita se asignará al usuario actual. Se mostrarán datos médicos en la pestaña Resultado, además se permitirá su edición
- Recepcionista: La pantalla presentará todas las visitas. Cuando se realice un alta no marcará ningún oftalmólogo obligando al usuario a seleccionar uno. La pestaña Resultado no mostrará datos y por tanto tampoco se podrán editar.

La pantalla la podemos dividir en diferentes áreas:



Área de selección

La misión del área de selección es establecer una serie de condiciones que determinarán que visitas se visualizarán en el área de visitas (que veremos más adelante). Se trata de un filtro y constituirá el Where de la Select sobre la tabla Visita. Se lanzará la consulta cuando pulsemos el botón “Mostrar según la selección”.

Controles del área:

Especifica desde que fecha serán mostradas las visitas.

Especifica hasta que fecha serán mostradas las visitas.

Especifica el paciente de las visitas, a la hora de construir el Where de la Select estos campos quedarán unidos por AND.

Al introducir un DNI especificamos que las visitas que se mostrarán en el área de visitas deberán haber sido realizadas al paciente con dicho DNI.

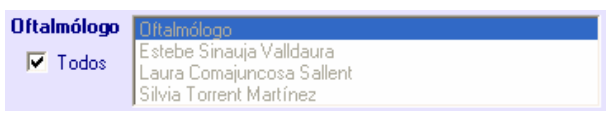
Permite localizar un paciente.

Muestra información del paciente actual (cuyo DNI sea el que haya introducido en ese momento).

Las visitas del paciente actual (cuyo DNI sea el que haya introducido en ese momento) serán mostradas en el área de visitas de forma inmediata y omitiendo el resto del filtro.

Especificamos que las visitas que se mostrarán deberán haber sido realizadas a un paciente cuyo nombre empiece por el texto que se introduzca en esta caja de texto. Internamente añade el carácter % al final del texto que introduzcamos. Es posible además utilizar comodines de SQL, por ejemplo poner al principio %, utilizar?, etc.

Especificamos que las visitas que se mostrarán deberán haber sido realizadas a un paciente cuyos apellidos empiecen por el texto que se introduzca en esta caja de texto. Internamente añade el carácter % al final del texto que introduzcamos. Es posible además utilizar comodines de SQL, por ejemplo poner al principio %, utilizar?, etc.

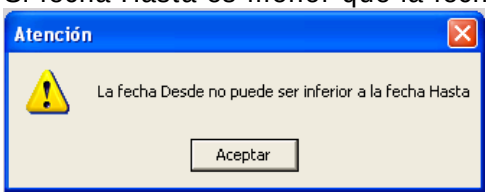


Permite especificar el oftalmólogo que ha realizado las visitas. Si el usuario actual es oftalmólogo por defecto aparecerá resaltado su nombre en este control y "Todos" desmarcado.

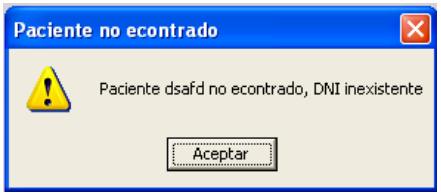
Una vez introducidos los datos del área de selección pulsaremos este botón para ejecutar la consulta.

A tener en cuenta:
Las cajas de texto vacías no intervienen en el Where.

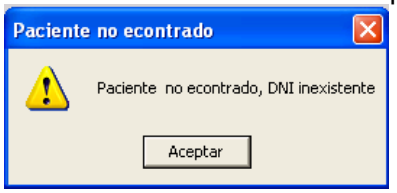
Si fecha Hasta es menor que la fecha Desde aparecerá el siguiente mensaje:



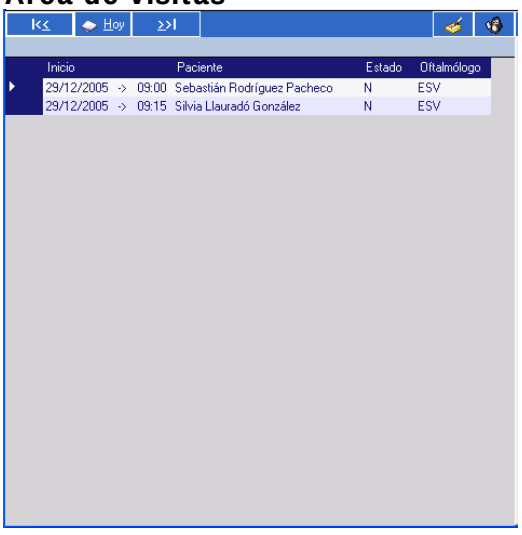
Si en los controles de DNI del paciente se introduce un DNI inexistente y se pulsa el botón información o modificación aparecerá el siguiente mensaje:



Si en los controles de DNI del paciente no se introduce un DNI y se pulsa el botón información o modificación aparecerá el siguiente mensaje:



Área de visitas



La misión del área de visitas es mostrar las visitas en función a lo establecido en el área de selección. El área de visitas cuenta con una serie de botones que permiten visualizar visitas en base a los criterios más comunes, sin necesidad de variar manualmente el área de selección, podrían denominarse botones de consulta rápida. Al pulsar sobre una fila se actualiza el área de la visita (que veremos en el próximo apartado), siempre y cuando no se esté llevando a cabo alguna operación (alta o modificación de una visita).

Controles del área:

Permiten al usuario moverse día a día con mucha facilidad

Visualiza las visitas del día anterior al día del calendario Desde (Área de selección). Lo hace estableciendo la fecha Desde y Hasta de la selección al día anterior.

Visualiza las visitas de hoy. Lo hace estableciendo la fecha Desde y Hasta de la selección al día de hoy

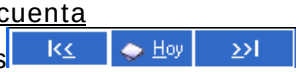
Visualiza las visitas del día posterior al día del calendario Desde (Área de selección). Lo hace estableciendo la fecha Desde y Hasta de la selección al día posterior.

Mediante estos botones el usuario puede visualizar las visitas en base a la visita seleccionada actualmente.

Imaginemos que estamos visualizando las visitas de 7 días simultáneamente y que estando situados en una de ellas nos gustaría ver tan sólo las visitas de ese día, pues la forma de conseguirlo sería pulsando este botón.

Imaginemos que tenemos seleccionada la visita del próximo paciente a visitar, sin duda sería interesante visualizar su historial. Al pulsar este botón aparecerán todas las visitas del paciente de la visita seleccionada.

A tener en cuenta

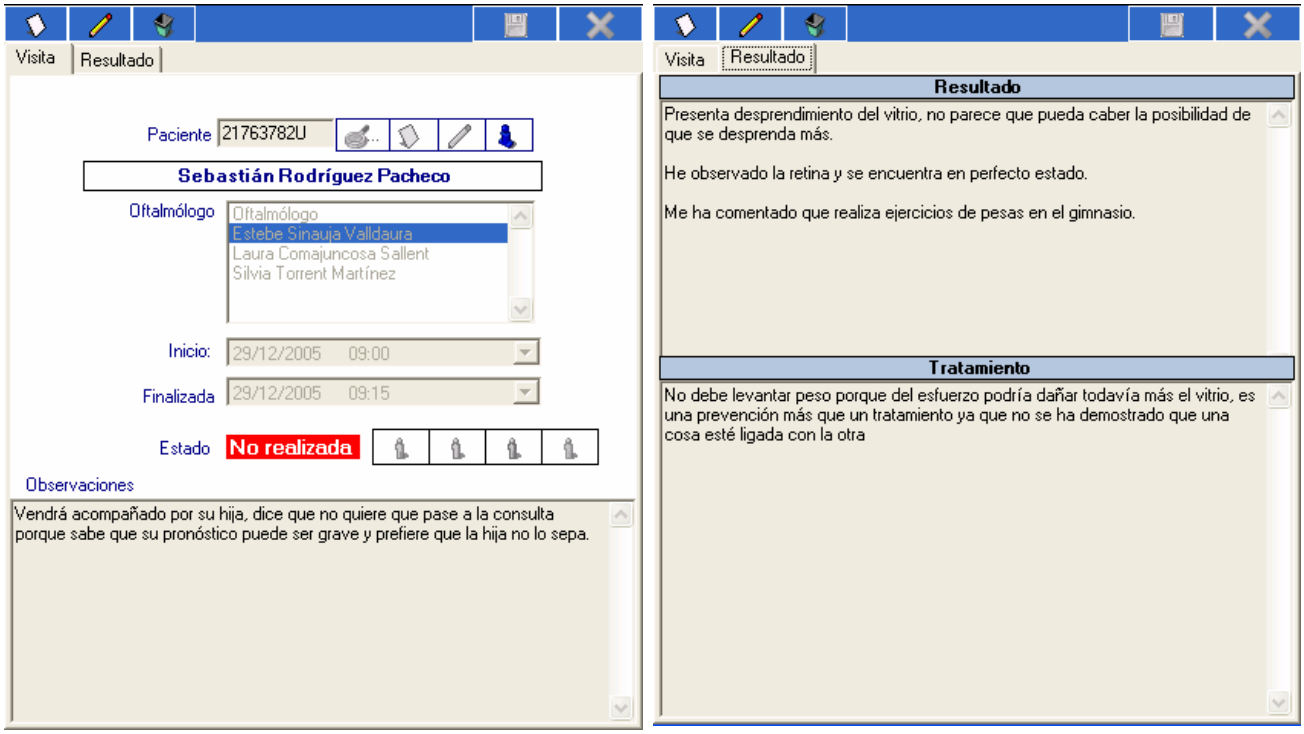
Los botones 

varían la fecha Desde y Hasta del área de selección y tienen en cuenta el resto de los datos introducidos en dicha área para mostrar los resultados (Datos del paciente y del oftalmólogo).

El botón  no altera el área de selección y tiene en cuenta los datos introducidos en dicha área para mostrar los resultados (a excepción de las fechas Desde y Hasta).

El botón  no altera el área de selección y omite los valores introducidos en dicha área.

Área de la visita

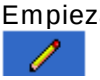


El área de la visita muestra la visita seleccionada en el área de visitas, permite también modificarla o eliminarla. También permite añadir nuevas.

Controles del área:

Mediante estos botones podemos llevar a cabo acciones en una visita en particular

Empieza el alta de una nueva visita.



Permite modificar la visita actual.

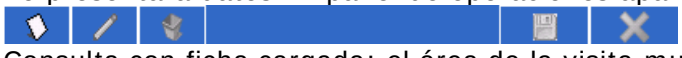
Elimina la visita actual. Sólo pueden eliminarse visitas no realizadas.

Guarda los cambios realizados y finaliza la operación, tanto si se trata de un alta como de una modificación.

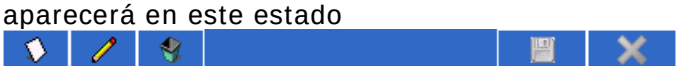
Cancela los cambios realizados, finalizando la operación, tanto si se trata de un alta como de una modificación.

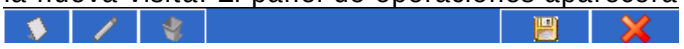
Posibles estados del área:

- Ficha no cargada: cuando el área de visitas no visualiza ninguna visita (La rejilla no tiene filas), el área de visita estará bloqueada y lógicamente no presentará datos. El panel de operaciones aparecerá de esta forma

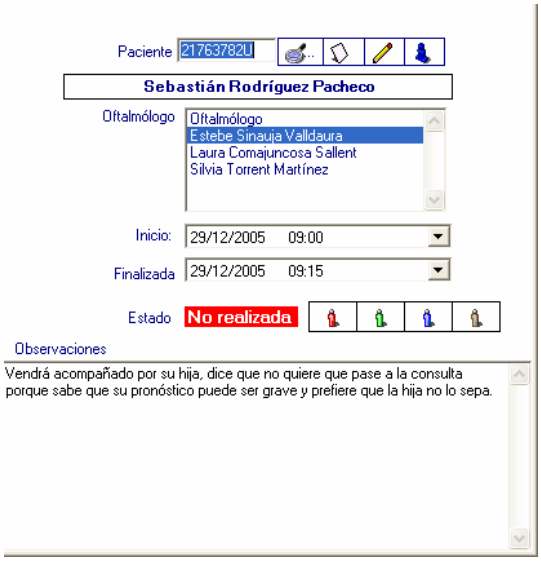


- Consulta con ficha cargada: el área de la visita muestra la visita seleccionada en el área de visitas. Es fácil distinguir dicho estado porque los controles aparecen bloqueados (a excepción del botón información del paciente) y contienen datos, además el panel de operaciones

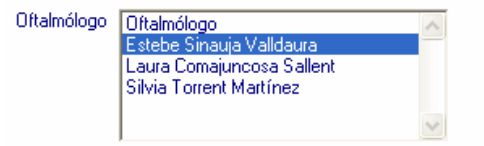
aparecerá en este estado 

- Modificación: Para encontrarnos en este estado es necesario haber pasado antes por el estado Consulta con ficha cargada. Podremos modificar los valores de la visita. El panel de operaciones aparecerá de la siguiente forma  a la espera de que aceptemos los cambios o los cancelemos.
- Alta: Al encontrarnos en este estado podremos introducir los valores de la nueva visita. El panel de operaciones aparecerá en este estado  a la espera de que aceptemos los cambios o los cancelemos.

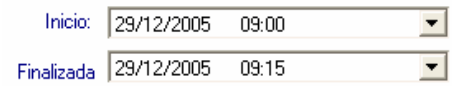
Los siguientes controles permiten introducir/modificar los datos de una visita así como visualizarla.



Permiten especificar el paciente de la visita, localizándolo  o introduciendo uno nuevo  . También permiten modificar  el paciente actual, por ejemplo para concederle acceso Web, también permiten ver la ficha del paciente actual  , para ver su información  . Si el DNI del paciente es correcto al abandonar la caja de texto veremos su nombre y apellidos



Permite especificar el oftalmólogo que ha llevado o llevará a cabo la visita. Si el usuario actual es oftalmólogo cada vez que inicie el alta de una nueva visita aparecerá su nombre resaltado en azul, tal y como podemos ver en la figura.

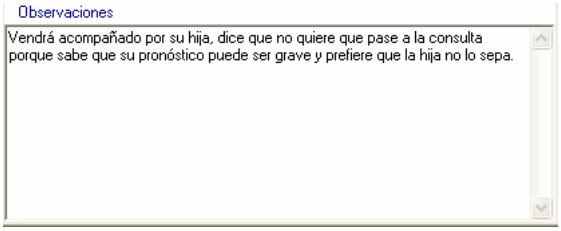


En estos controles se especifica la fecha y hora de inicio de la visita y la fecha y hora de finalización. Estos controles funcionan conjuntamente, al variar la fecha y hora de inicio automáticamente se varía también la de finalización (añade un cuarto de hora a la fecha hora de inicio).

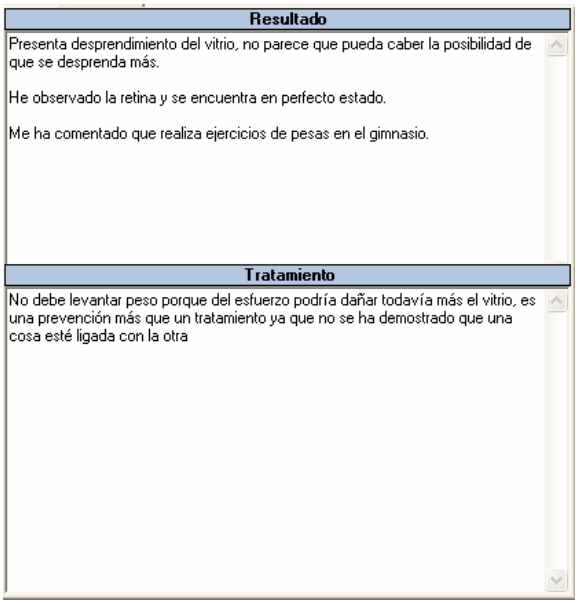
*Porque aparece fecha de finalización? Porque si el centro oftalmológico tiene horario de urgencias puede realizar visitas por ejemplo de 23:55 a 0:05 del día siguiente.

Mediante estos controles se informa y establece el estado actual de la visita, a continuación incluyo una tabla indicando para cada botón que estado establece.

	No realizada
	Presentad@
	Se realizó
	No se realizó



Se introduce en este control las observaciones que consideremos oportunas.



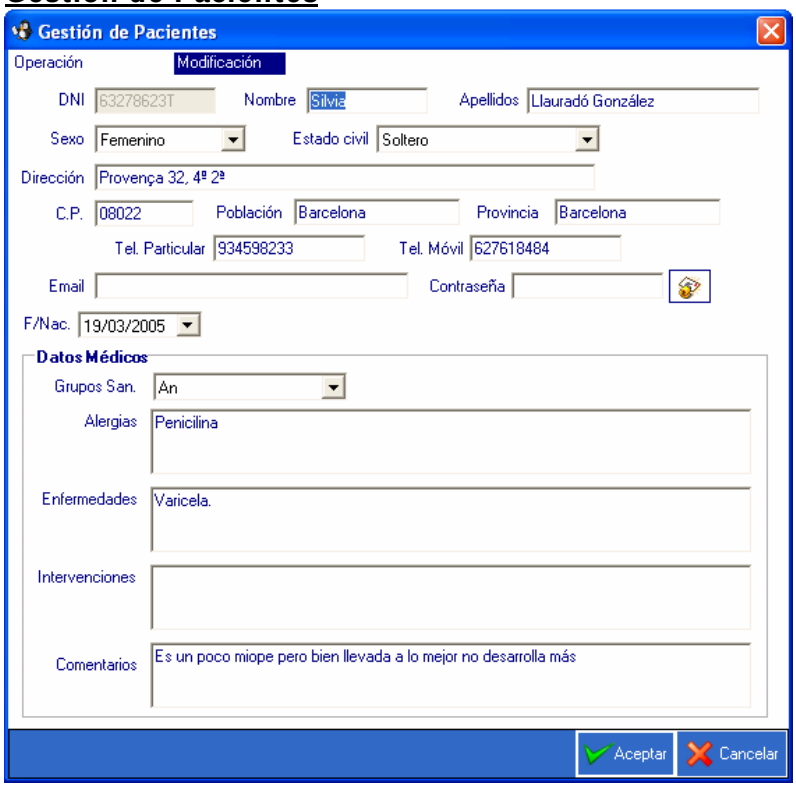
El oftalmólogo podrá introducir el resultado y tratamiento de la visita en estos controles. Recordemos que el recepcionista no podrá ver estos datos.

A tener en cuenta:

*******Importante: Mientras se modifica o introduce una nueva visita, al cambiar la fecha de inicio o el oftalmólogo que la realiza, el área de visitas**

cambiará mostrando resultados en base a lo que estemos introduciendo.
Por ejemplo, si el recepcionista esta dando de alta una visita para el día 3 de febrero, al buscar dicha fecha en el calendario de la fecha inicio verá en el área de visitas las visitas del día 3 para saber si puede dar hora o no, obviamente del oftalmólogo que tenga seleccionado en ese momento, si selecciona otro oftalmólogo se variará el área de visitas mostrando las visitas del nuevo oftalmólogo para ese día.

Gestión de Pacientes



La gestión de pacientes permite ver, modificar e introducir la información de un paciente. La pantalla es visualizada desde la gestión de visitas y tiene dos modos de presentación en función de si el usuario actual es un:

- Oftalmólogo: Aparecen los datos médicos tal y como vemos en la figura.
- Recepcionista: No aparecen los datos médicos, tal como podemos ver en la siguiente figura

Las operaciones que podemos efectuar son:

- Alta: Realiza el alta de un nuevo paciente. Cuando la pantalla está en operación alta el botón  aparece desactivado y si incluimos el email del

paciente y una contraseña de acceso al aceptar el paciente se enviará de forma automática un email con los datos de acceso.

- Modificación: Realiza la modificación de un paciente. No permite cambiar el DNI del paciente. Cuando la pantalla está en operación modificación el botón  aparece activado permitiendo enviar al paciente los datos de acceso si lo consideramos oportuno.
- Consulta: Permite consultar los datos de un paciente. Cuando la pantalla está en operación consulta los controles aparecen deshabilitados. Cuando la pantalla está en operación modificación el botón  aparece activado permitiendo enviar al paciente los datos de acceso si lo consideramos oportuno.

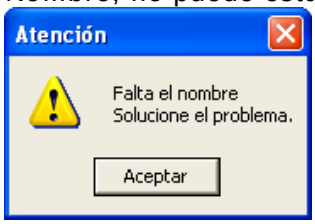
El email con los datos de acceso es el siguiente

A tener en cuenta:

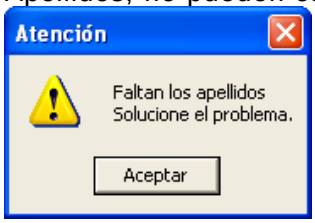
En operación alta se comprueba que los datos sean correctos, en concreto se comprueba:

DNI, no puede estar vacío y debe tener un ancho de 9, de lo contrario mostrará el mensaje

Nombre, no puede estar vacío, de lo contrario se mostrará el mensaje:



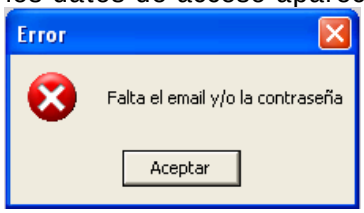
Apellidos, no pueden estar vacíos, de lo contrario se mostrará el mensaje:



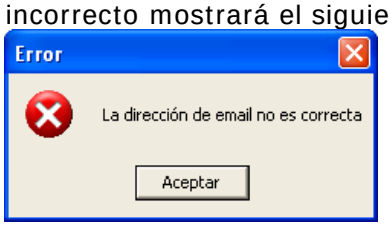
Teléfonos, como mínimo debe haber uno introducido, de lo contrario se mostrará el mensaje:

La pantalla indica en todo momento la operación que está llevando a cabo mediante tres etiquetas:

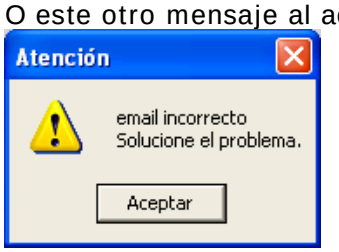
Si no se introduce e-mail y/o contraseña al pulsar el botón de envío del email con los datos de acceso aparecerá el siguiente mensaje



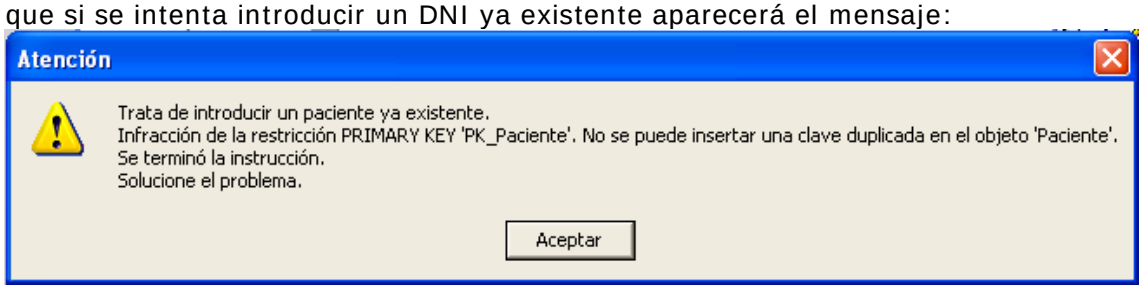
Se ha utilizado una “RegularExpression” para validar el email, por lo que si es incorrecto mostrará el siguiente mensaje al intentar enviar el mensaje



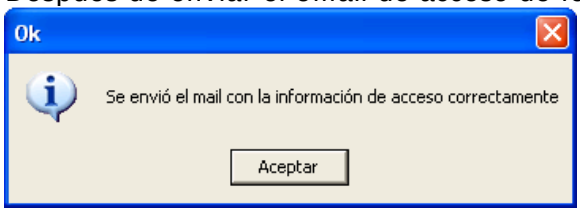
O este otro mensaje al aceptar



Se intercepta la excepción por intento de duplicación de la clave principal, por lo que si se intenta introducir un DNI ya existente aparecerá el mensaje:



Después de enviar el email de acceso de forma correcta aparece el mensaje



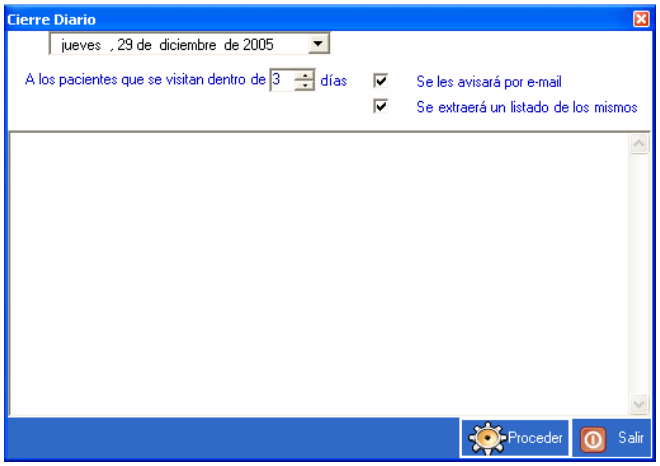
Consulta de Pacientes

La consulta de pacientes nos ayuda a localizar un paciente determinado. Es llamada desde la Gestión de visitas, para localizar el paciente de la visita.

Como podemos observar en la figura hay una serie de criterios de selección que formarán el where de la select contra la tabla Paciente. Dichos criterios quedarán unidos por un AND.

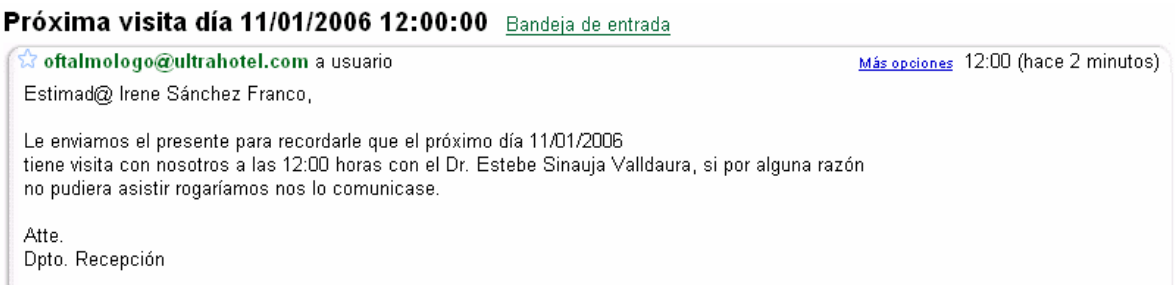
En los campos Nombre y Apellidos internamente se añade el carácter % al final del texto que introduzcamos. Es posible además utilizar comodines de SQL, por ejemplo poner al principio %, utilizar ?, etc.

Pantalla de cierre

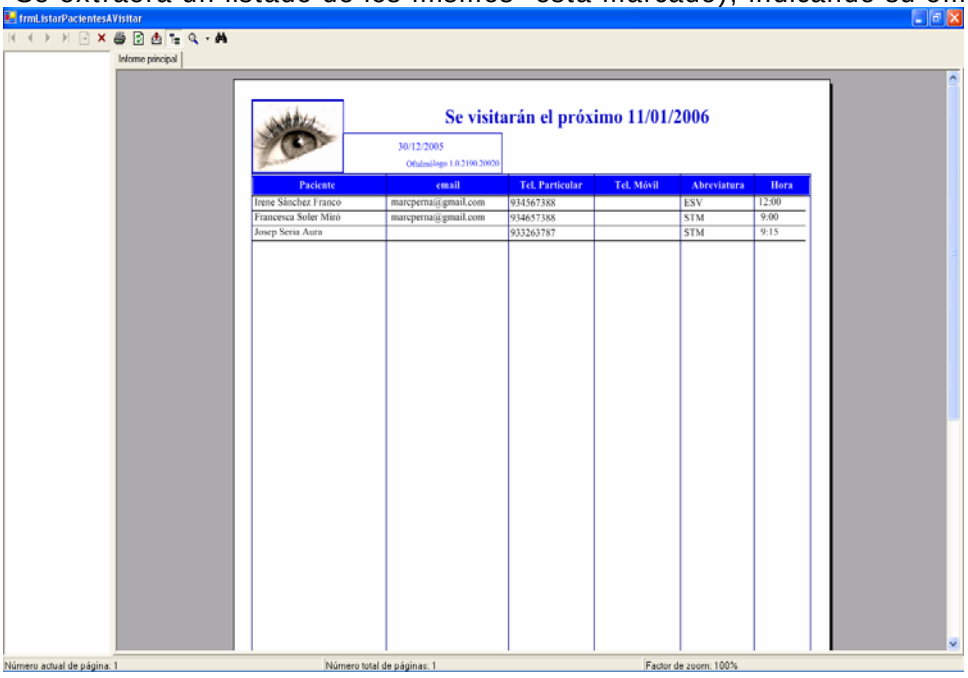


La pantalla de cierre finaliza el día de la consulta, para ello efectúa las siguientes operaciones:

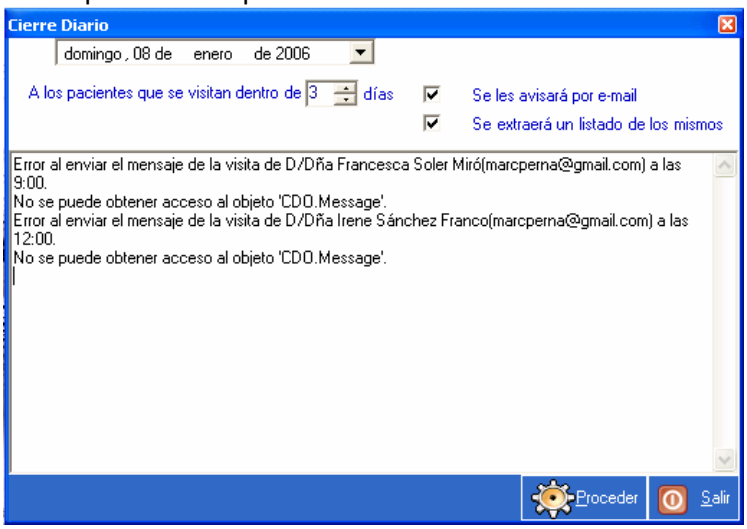
1. Marca las visitas en estado “No realizada” a estado “No se realizó”
2. Calcula la fecha de aviso en base a lo introducido en el número de días
3. Avisa a los pacientes (Si “Se les avisará por e-mail” está marcado) que se visitarán en esa fecha de aviso mediante un email similar a éste:



4. Extrae un listado de los pacientes que se visitarán en la fecha de aviso (Si “Se extraerá un listado de los mismos” está marcado), indicando su email:

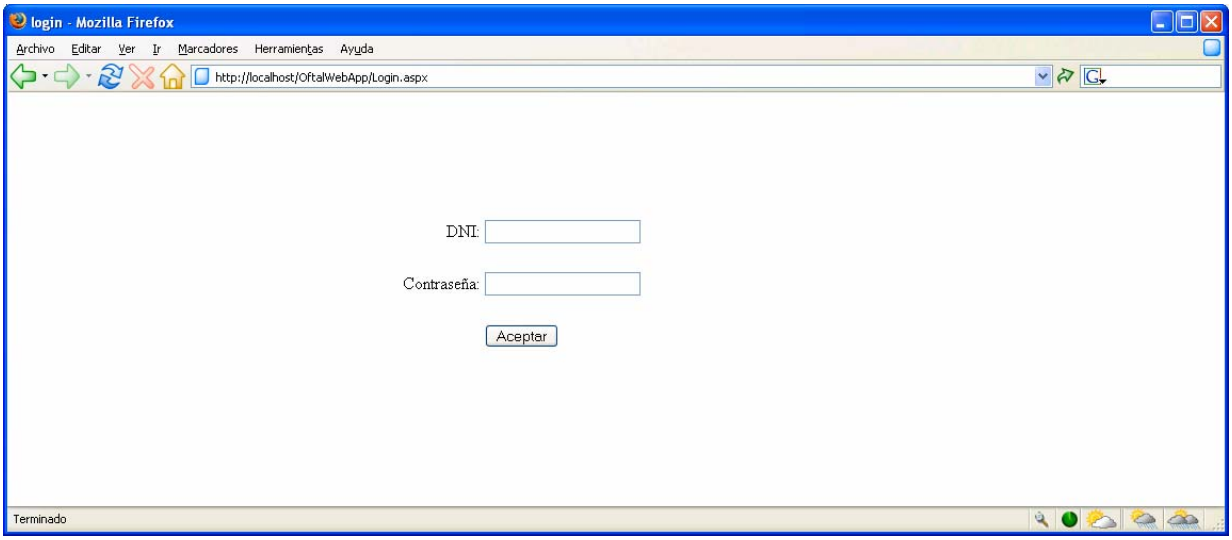


5. Si hubiera habido problemas al enviar emails de aviso de visita lo indicará en la pantalla de proceso

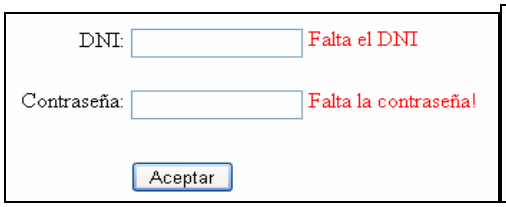


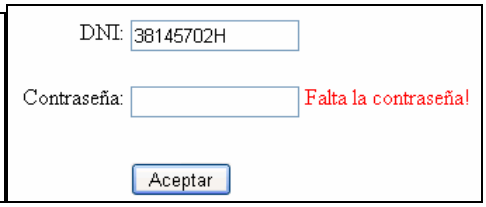
6. 4.- Subsistema del paciente

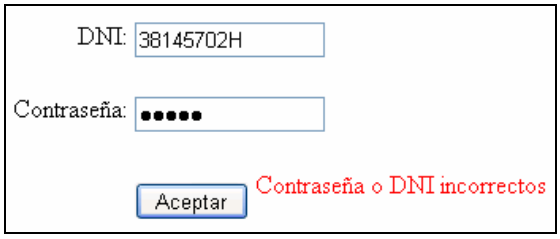
Login o pantalla de entrada



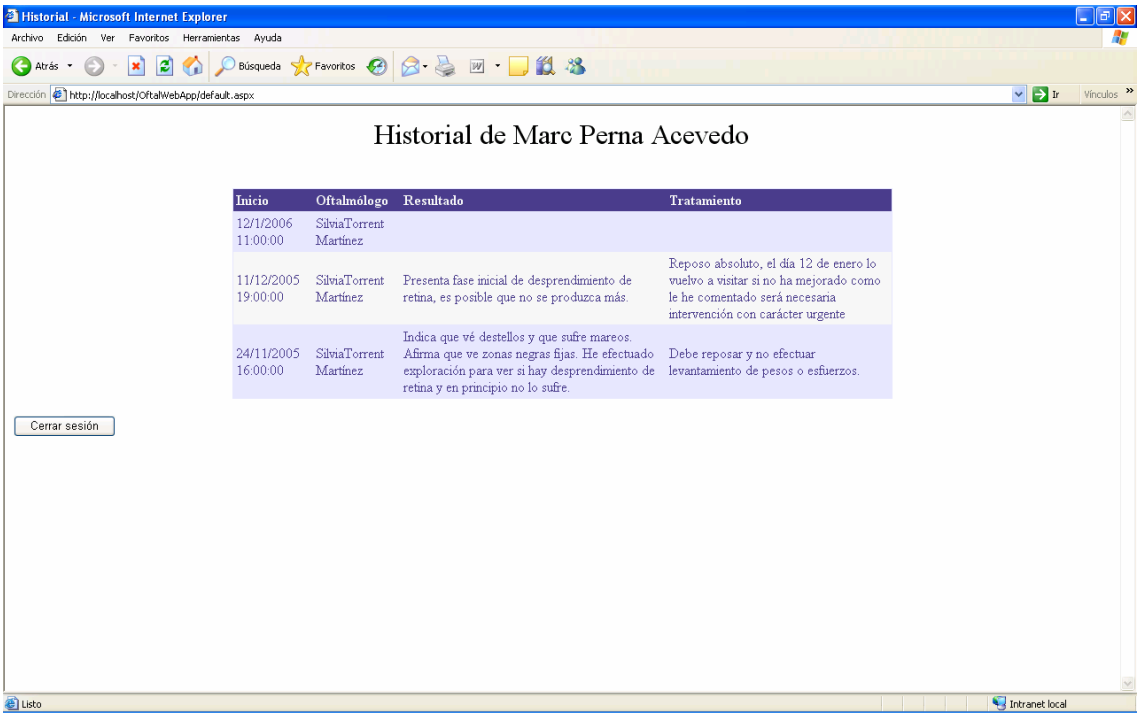
La pantalla de entrada permite al paciente conectarse a nuestra Web para acceder a su historial, realiza una serie de comprobaciones tal y como puede verse en las siguientes figuras







Pantalla de Historial



La pantalla de historial muestra el historial al paciente, de esta forma puede ver sus visitas, resultados, tratamientos y cuando se tiene que visitar.

Se permite cerrar sesión ya que se trata de datos médicos confidenciales

Conclusiones

Este trabajo ha aportado un enfoque diferente a la mayor parte de aplicaciones ya que todas se centran en el escritorio o en la Web. Este proyecto se ha centrado en el escritorio ciertamente pero sin renunciar a la Web y ha alcanzado todos los objetivos planteados al principio, incluso superándolos ampliamente. Ha integrado en la gestión mecanismos automáticos de aviso, como el envío de emails integrándolos como parte normal de la gestión que lleva a cabo el software. También ha tocado el área de diseño de informes ya que indudablemente en una aplicación normal se utilizan continuamente. Se han utilizado toda clase de controles, aunque debo decir que los que trae .Net por defecto son todavía un poco pobres, en especial el DataGrid.

Se ha cuidado mucho el aspecto gráfico de la aplicación así como su usabilidad y robustez.

Se ha cuidado muchísimo el diseño y el análisis, se ha ido incluso un poco más allá ya que este proyecto no sólo se ha limitado a llegar a las funcionalidades necesarias, ha querido además consolidar una base para que más adelante pueda crecer e incluso para reaprovechar su base en otros proyectos.

También debo decir que he alcanzado un alto conocimiento de .Net que era uno de los objetivos principales.

Un proyecto siempre puede mejorar y podría no cerrarse nunca, seguro que siempre hay más funcionalidades que implementar, nuevas pantallas que hacer y lógicamente nuevos requisitos que irán apareciendo con el tiempo, pero lo importante siempre es tener una buena base que te permita evolucionar la aplicación y creo que este proyecto lo ha conseguido.

Líneas futuras de trabajo

Este proyecto ha sido desarrollado en un tiempo muy corto, y por tanto se han cubierto las necesidades básicas de un centro oftalmológico pero hay muchas más que pueden desarrollarse en el futuro, como:

- Parte Windows
  - Horarios de oftalmólogos y control a la hora de realizar el alta de nuevas visitas
  - Permitir configurar que un oftalmólogo pueda no estar visible en la pantalla de Visitas
  - Operaciones
  - Planning de operaciones por oftalmólogo
  - Gestión de mutuas
  - Facturación (Visitas, intervenciones, medicamentos, …)
  - Gestión de cobros
  - Aparatos médicos
  - Amortizaciones de aparatos
  - Stock de medicamentos
  - Ratios de producción
  - Contabilidad
  - Etc.
- Parte Web
  - Permitir a los oftalmólogos consultar el historial de un paciente vía Web
  - Permitir introducir resultados de una visita vía Web. Útil si el oftalmólogo debe desplazarse.
  - Permitir a los usuarios registrados pedir visita por Web
  - Ver los pagos realizados por una mutua, o de los pagos pendientes de realizar. De esta forma las mutuas podrían conectarse a nuestra Web a ver su estado.
  - Etc.

Pero al margen de que el software siempre puede mejorar y necesitar nuevas funcionalidades lo bueno de este proyecto es que ya tiene la base establecida ya que ha incorporado:

- Jerarquías de clases de acceso a datos
- Jerarquías de clases gestoras
- Jerarquías de clases gráficas
- Clases utilitarias
- Usuarios
- Grupos de usuarios
- Permisos de usuarios y/o grupos sobre menús
- Parte Web
- Etc.

Un proyecto siempre puede mejorar y podría no cerrarse nunca, seguro que siempre hay más funcionalidades que implementar, nuevas pantallas que hacer y lógicamente nuevos requisitos que irán apareciendo con el tiempo, pero lo importante siempre es tener una buena base que te permita evolucionar la aplicación, y este proyecto ha seguido fielmente este objetivo.

Bibliografía

Professional C# 2ª Edición (wrox)
Diseño de aplicaciones con Microsoft ASP .net – Douglas J. Reilly (Mc Graw Hill)
Creación de servicios web Xml para la plataforma Microsoft .Net – Scout Short (Mc Graw Hill)

Webs

Uso de controles, ayudas y ejemplos
http://msdn.microsoft.com/
http://www.c-sharpcorner.com/
http://www.codeproject.com/csharp/
http://www.developerfusion.co.uk/csharp/

Lenguaje, clases y ejemplos
http://msdn.microsoft.com/
http://www.developerfusion.co.uk/csharp/
http://www.codeproject.com/csharp/
http://www.lawebdelprogramador.com/codigo/mostrar.php?id=237&texto=Visual+CSharp+.NET
http://www.developerland.com/CuttingEdge/VS2005/262.aspx

Información de la plataforma, futuras versiones y mejoras
http://msdn.microsoft.com/
http://www.csharp-station.com/

Patterns & Practices
http://www.microsoft.com/spanish/msdn/comunidad/mtj.net/voices/MTJ_3317.asp
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpatterns/html/DesObserver.asp
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnpatterns/html/DesMVC.asp
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/dnbda/html/apparchch1.asp
http://msdn.microsoft.com/practices/

Foros de ayuda
http://www.msusenet.com/forumdisplay.php?f=13

Testeo de aplicaciones
http://www.nunit.org/

DataGrid
http://216.239.51.104/search?q=cache:C8moReFrSI4J:www.morpheusweb.it/html_eng/scripts/csharp/csharp_sessionicookie.asp+read+cookie+.net+c%23&hl=es&client=firefox-a
http://www.syncfusion.com/FAQ/WindowsForms/FAQ_c44c.aspx#q1096q

Cookies, SessionState, Authentication
http://www.morpheusweb.it/html_eng/scripts/csharp/csharp_sessionicookie.asp
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconsessionstate.asp
http://msdn.microsoft.com/library/default.asp?url=/library/en-us/cpguide/html/cpconTheWindowsAuthenticationProvider.asp
http://support.microsoft.com/default.aspx?scid=kb;es;301240

Bases de datos
http://www.fabforce.net/dbdesigner4/screenshot_image.php?screenshot=dbd4_ss_simplemodel.png